KONTEKSTUELL KLASSIFISERING AV BILDER FOR Å KARTLEGGE AKVATISK VEGETASJON

av

ROAR BRÆNDEN

MASTEROPPGAVE
for graden
Master i Modellering og dataanalyse

(Master of Science)

Det matematisk- naturvitenskapelige fakultet
Universitetet i Oslo

April 2010

Faculty of Mathematics and Natural Sciences
University of Oslo

Forord

Denne oppgaven markerer avslutningen av et langt og variert studium ved Universitetet i Oslo. Siste del har foregått på deltid samtidig som jeg har fulgt opp en fast stilling hos Norsk institutt for vannsforskning (NIVA). Disse har bidratt med bildene som brukes i denne oppgaven, og de har bidratt med prosjektmidler slik at jeg kunne gjennomføre deler av oppgaven i arbeidstida. Takk til Marit Mjelde og John Rune Selvik for deres bidrag.

Så vil jeg takke mine veiledere Ingrid Glad og prof. Nils Lid Hjort for gode innspill. Personlig har mitt fokus vært den praktiske tilnærmingen, og de har jobbet hardt for å få inn en teoretisk tilnærming. Dette vil nok oppgaven bære preg av.

Til slutt vil jeg takke Kristin for hennes støtte og oppmuntrende ord, og at jeg har fått mulighet til å bruke så mye tid på disse studiene.

Oslo, april 2010

Roar Brænden

Innhold

1 Innledning

Vegetasjon i innsjøer, strandkanter og elver går under fellesbetegnelsen akvatisk vegetasjon. Her finnes det ulike former, noen lever helt under vann, andre dupper i vannskorpa mens atter andre har røttene i strandkanten mens resten av planten står godt over vann. Dette er typisk for siv-artene, som også kalles makrofytter.

Omfanget av denne vegetasjonen blir ofte brukt som en indikator på miljøtilstanden i innsjøen eller elva. Dersom det er stor næringstilgang vil vi få en oppblomstring av vegetasjon. Fordelingen mellom de ulike vegetasjonstypene kan brukes som en indikasjon på fordelingen mellom ulike næringsstoffer.

For å kunne utnytte denne indikatoren er vi avhengig av å ha en god kartlegging av vegetasjonen gjennom flere år. Enhver innsjø og elv vil ha sin egen naturlige sammensetning av vegetasjon. Dette er et resultat av de lokale geologiske og klimatiske forholdene. I tillegg kommer menneskeskapte påvirkninger ved jordbruk, industri eller boligutbygging. Det er endringer i denne sammensetningen man som regel er på jakt etter.

Kartlegging av akvatisk vegetasjon byr på en rekke utfordringer. I forbindelse med kartlegging av innsjøer er det mulig å bruke båt, men det er ikke alltid like enkelt å komme til innsjøene med båt. Og det er sjelden noe godt virkemiddel når man skal kartlegge elver. Derfor er det ofte ønskelig å komme seg opp i lufta for å få bedre overblikk.

Dette kan man enten gjøre ved å bruke flyfotografier, eller man kan bruke bilder tatt fra satellitt. I begge tilfeller får man et bilde, og så blir oppgaven å trekke informasjon ut av dette som kan tilbakeføres til de naturlige forholdene. Dette kalles fjernanalyse.

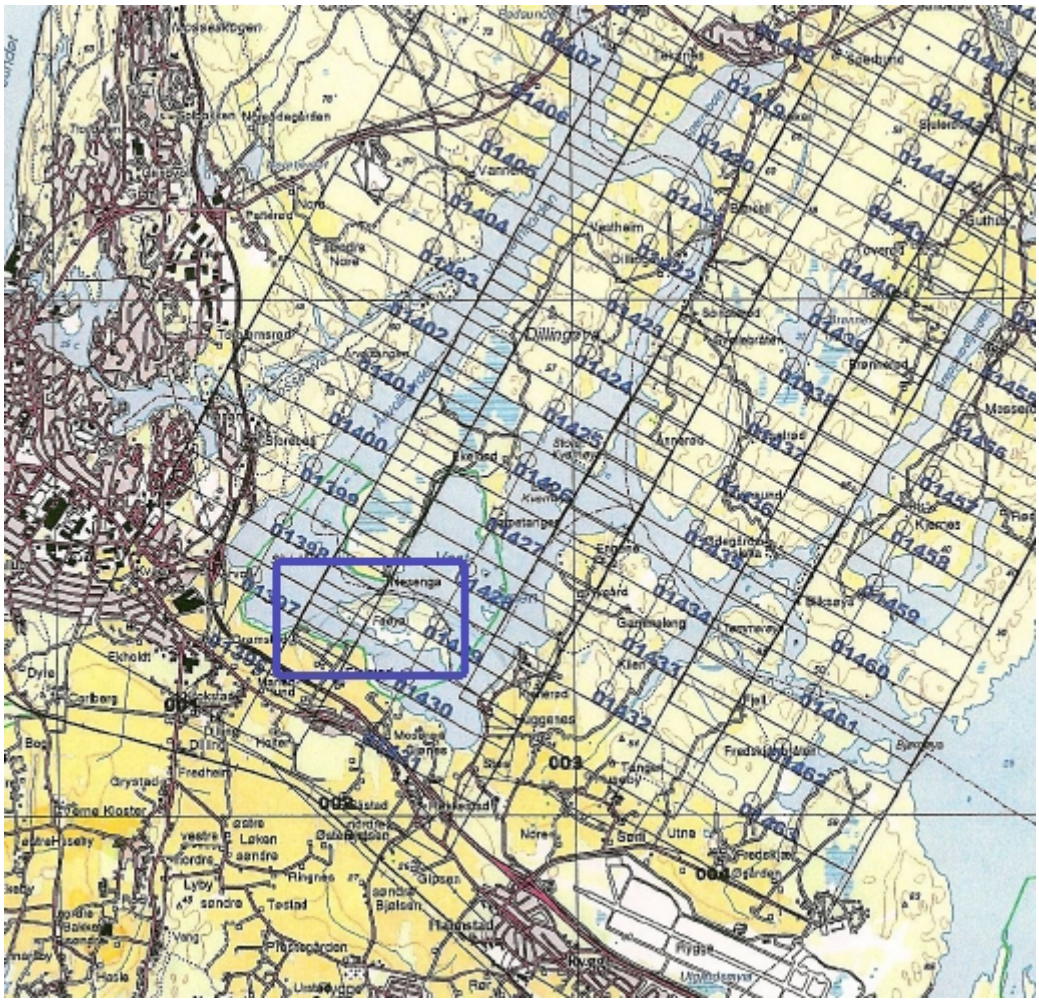
I fjernanalyse opererer man med optiske eller aktive sensorer. Optiske sensorer fanger opp sollyset som reflekteres fra overflaten. Dette kan både være synlig lys slik vi registrerer med øynene våre, eller elektromagnetisk stråling i andre frekvensområder. Med aktive sensorer menes sensorer som selv sender ut en elektromagnestik stråling, fanger opp refleksjonen av denne og analyserer differensen.

Når vi har et opptak skal det bearbeides for å trekke ut ønsket informasjon. Dersom vi har et bilde basert på synlig lys vil et trenet øye kunne tolke mye ved kun å se på bildet, men som regel vil det være tidsbesparende å benytte seg av en statistisk modell for å tolke hele bildet. En slik modell tar utgangspunkt i fargeverdiene i hvert enkelt piksel. Disse tolkes som uavhengige variabler. For hver

piksel vil vi ha flere verdier, som svarer til hver sin farge eller kanal.

En vanlig modell går ut på å gruppere pikslene i klasser avhengig av fargeverdiene. Hver klasse representerer en landskaps- eller vegetasjonstype. Alle pikslene innen en gruppe fargelegges i en egnet farge, og på denne måten får man frem et landskaps- / vegetasjonskart. Grunnlaget for en slik modell vil som regel være et treningssett med de ønskede gruppene. Dette bygges opp ved at en person markerer i bildet områder han gjenkjenner som en av de vegetasjons- eller landskapstypene man ønsker å kartlegge.

I denne oppgaven skal vi benytte oss av et et sett med flyfotografier, som ble tatt opp høsten 2005 i regi av NIVA. Fotografiene dekker et område over Vansjø i Østfold. Kameraet som ble benyttet registrerer optisk lys i kanalene infrarødt, rødt og grønt.



Figur 1: Flyfotoene som brukes i denne oppgaven er markert med blå firkant. Kartet viser forøvrig Moss til venstre og Rygge flystasjon nede til høyre.

Hovedfokus i denne oppgaven vil være bildet med betegnelsen 01429. Dette dekker øya Feøya med tilliggende områder. Her har NIVA gjennom en årrekke gjort feltobservasjoner for å kartlegge vegetasjonen. I tillegg vil jeg i kapittel 8 bruke bildet 01398, som overlapper med 01429.

1.1 Oppbygning av oppgaven

Kapittel 2 og 3 gir en teoretisk bakgrunn for henholdsvis fjernanalysen og biologien. I kapittel 4 vil jeg se på ulike algoritmer som brukes innenfor statistisk klassifisering. Først vil jeg ta for meg de som utelukkende baserer seg på fargeverdiene, mens i kapittel 5 vil jeg utvide modellen ved også å se på naboforhold, eller konteksten pikselet opptrer i.

Begge disse kapitlene vil være basert på bilde 01429, og et treningssett som er basert på en manuell gjennomgang av bildet. I kapittel 6 vil jeg se nærmere på sammenhengen mellom den ikke-konteksuelle og kontekstuelle klassifiseringen, men denne gangen i et fiktivt bilde.

I kapittel 7 vil jeg se mer på treningssettet og betraktninger rundt dette. Mens jeg i kapittel 8 ser på mulighetene for å bruke samme gruppeinndeling og parametersett på flere bilder i samme serie. Et flytokt over et vassdrag som Vansjø utgjør mange bilder. Dette fremkommer av figuren på forrige side. Det ville vært en stor fordel om man kunne begrenset den manuelle kartleggingen til et utvalg av disse bildene. For deretter å overlate til datamaskinen å regne seg gjennom de resterende bildene.

1.2 Notasjon

Gjennom oppgaven vil y betegne de observerte verdiene i hele bildet. Når den er indeksert y_i indikerer det at vi ser på verdiene i ett enkelt piksel. Når jeg bruker i som indeks fremfor f.eks i,j, så er dette for å forenkle notasjonen, men det ligger en implisitt konvertering til bildets rutenett. Selv om denne ikke er av betydning for denne fremstillingen, har jeg gjennomgående startet i øvre venstre hjørne og tatt rad for rad.

Klassetilhørigheten betegnes med x. Dette er en diskret variabel som tar verdi i et sett bestående av K klasser. Tilsvarende som for y vil x_i betegne klassetilhørigheten for ett enkelt piksel.

2 Fjernanalyse

Fjernanalyse omfatter opptak og analyse av bilder tatt fra en plattform over bakkenivå. I våre dager vil plattformen enten være fly, helikopter eller satellitt. Bildene må forstås i en bred sammenheng. Det kan være tradisjonelle fotografier, slik de blir presentert på gulesider.no eller Google Earth, men det kan også være radarbilder som brukes av meteorologene til å kartlegge omfanget av et nedbørsområde. I begge tilfeller er bildet en representasjon av den elektromagnetiske strålingen som treffer en sensor. Sensorene er konstruert slik at de reagerer på ståling innenfor ulike frekvensområder.

På samme måte er våre øyne utstyrt med en mengde reseptorer, som kan deles i tre grupper. Hver gruppe reagerer på elektromagnetisk stråling innen hvert sitt frekvensområde. Dette gir grunnlag for fargene rødt, grønt og blått, og intensiteten i hvert av disse frekvensområdene er bestemmende for den fargen vi oppfatter.

I sammenheng med fjernanalyse og undersøkelser av naturgrunnlaget er det ikke nødvendigvis disse frekvensene som er av størst betydning. Ofte er det ønskelig å se på refleksjonen av infra-rødt, fordi dette er stråling som egner seg godt til å skille vegetasjon fra andre landskapsformer. Derfor er satellitter som Landsat, Spot utstyrt med sensorer som måler elektromagnetisk stråling i disse frekvensområdeneer aktiv i fotosyntese.

I tillegg til plattform og sensor har vi valg av lyskilde. Det vanligste er å bruke sola, og registrere det som reflekteres fra bakken. Dette omtaler vi som passive sensorer. Tilsvarende kan vi også bruke aktive sensorer. Disse sender ut elektromagnetisk stråling av en gitt frekvens og registrerer det som returneres. Det er en slik som brukes av meteorologene for å lage bilder av nedbørsområdenes omfang.

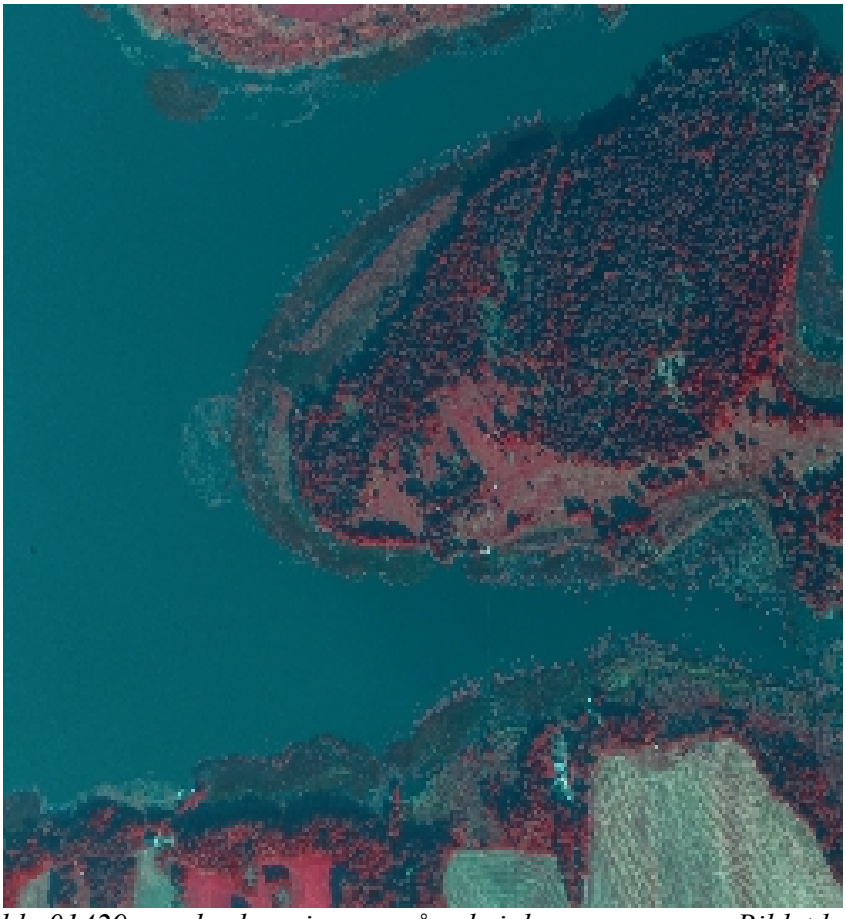
2.1 Digital representasjon

En passiv sensor inneholder optikk som konsentrerer den elektromagnetiske strålingen ned på en liten plate. Denne er delt inn i et finmasket rutenett, hvor hver rute gjerne omtales som et piksel (picture element). I platen er det elektriske komponenter som registrerer den innkommende strålingen som en elektrisk ladning. Denne ladningen leses av for hvert piksel i platen, og registreres som et heltall.

Sensorene kan operere med ulike oppløsninger på heltallet de registrerer. Inntil nylig har den vanligste oppløsningen vært 8 bit. Dette svarer til at man har et heltall i skalaen 0-255. Nyere sensorer kan ha en oppløsning på 16 bit. Da opererer vi på skalaer fra 0-2047. Dette innebærer at man for hver scene får en betydelig større datamengde. Her vil det alltid være en avveining om dette er ønskelig i forhold til den oppgaven bildet skal brukes i. Blant annet må det sees i lys av den programvaren som skal brukes for å anaysere bildene. Det er ikke alle av disse som håndterer 16 bits kanaler, og da ender man opp med å nedskalere verdiene.

Det opptaket vi har tilgjengelig hadde i utgangspunktet en oppløsning på 16 bit for hver kanal, men fordi dette ga unhåndterlige filstørrelser valgte jeg å skalere disse ned til en oppløsning på 0-255 for hver kanal.

Hver bildefil som ble tatt hadde 15000*7500 piksler. Dette dekker et område på ca. 3*1,5 km. Jeg valgte å fokusere på et uttrekk av dette området. Dette består av 4000*4000 piksler, eller 16 millioner piksler. Det viste seg at dette var litt uhåndterbart, så jeg valgte å nedskalere ved å bruke hvert 16. piksel. Da har jeg en bildestørrelse på 250*250 piksler, og denne er håndterbar. Her er bildet som brukes gjennomgående i oppgaven.



Figur 2: Bilde 01429 som brukes gjennomgående i denne oppgaven. Bildet kan være litt vanskelig å tyde fordi vi bruker infra-rødt. Dette fremkommer som rødt og er mest fremtredende i områder med vegetasjon. Det grønne er vann, mens de gulhvite områdene nede til høyre er jordbruksområder.

I dette bildet er det en fargeforskyvning fordi vi skal ha frem den infra-røde strålingen. Denne verdien er lagt inn i den rød kanalen i bildet. Dermed er de synlige fargene også forskjøvet. Det som fremstår som rødt i bildet er vegetasjon. Det som er blå-grønt er vann, mens det gulhvite nede i venstre hjørne er jordbruksområder.

2.2 Geometrisk korreksjon

Ved fjernanalyse fra fly tar man opptak i parallelle striper. Mellom stripene legger man gjerne inn et overlapp, og også langsetter stripen har man overlapp. Overlappet kan være så stort som 50 – 65% Lillesand et al. (1987). Dette kan vi få en antydning av i oversiktskartet i figur 1.

Det er flere grunner til at man bruker overlapp. Man er avhengig av det i forbindelse med kartlegging av topografi. Dersom vi legger de to bildene oppå hverandre og justerer dem slik at vi får mest mulig overlapp, vil eventuelle høydeforskjeller i terrenget fremkomme som forskyvninger mellom bildene. Dette kan man utnytte til å tegne inn høydekurver. Enten ved å bruke et stereoskop, eller med datamaskinen og egnet programvare.

Dette kan også utnyttes ved kartlegging av vegetasjon. Ved at det kan være enklere å tolke bildene dersom man får frem dybdevirkningen. For å få til dette er man avhengig av å bruke stereoskop, eller en form for 3-d briller.

En annen fordel med å ha overlappende bilder er at vi kan se terrenget fra ulike vinkler. Dermed unngår vi at noe blir liggende i skyggen av trær eller koller. I forbindelse med vann kan vinkelen med sola ha stor betydning for refleksjonen. Dersom sola treffer vannet i riktig retning i forhold til sensoren kan hele bildet bli ødelagt av solglimt fra vannoverflaten. Dette vil vi se et eksempel på i kapittel 8. Her tar vi for oss bildet 01398, som er overlappende med bilde 01429.

Over beskrev jeg en manuell operasjon for å sammenstille to bilder. Denne operasjonen svarer til en affin transformasjon. I kapittel 8 vil jeg vise hvordan jeg kommer frem til parameterene i den affine transformasjonen ved å lukke ut kontrollpunkter i de to bildene.

Dette er en enkel metode for geometrisk korrigering. Toutin (2007) tar for seg ulike metoder som brukes innenfor geometrisk korrigering av bilder tatt opp fra satellitt. Her vil vi ofte korrigere bildene i forhold til en geografisk referanseramme, slik at de kan relateres direkte til et kart el. likn.

Fremgangsmåten er at man finner punkter i bildet, som kan relateres til lokaliteter på bakken. Posisjonen for disse hentes enten fra et kart, eller ute i felten med en GPS, og så regner man seg frem til en transformasjonsmodell. Ofte vil det også være nødvendig å korrigere for høydeforskjeller i bildet. Dette gjør man ved å bruke en høydemodell.

3 Akvatisk vegetasjon

Akvatisk vegetasjon er en fellesbetegnelse for vegetasjon som vokser i vann. Denne vegetasjonen brukes ofte som en indikator på eutrofiering, eller at tilførslene av næringsstoffer til vassdraget er i ulage. Dette kan skyldes kloakktilførsler eller tilførsler fra landbruket. Et annet forhold ved akvatisk vegetasjon er at det kan skape problemer ved bruken av vassdraget. Dette gjelder i forbindelse med fiske, bading og annet friluftsliv. Et tilgrodd vassdrag er sjelden ønskelig.

I forbindelse med spørsmål om tilgroing oppstår det ofte et behov for å kartlegge utbredelsen av akvatisk vegetasjon i hele vassdraget. I Norge består et vassdrag gjerne av mange småvann, tilknyttet med elver og bekker. Å kartlegge dette med båt kan være en tidkrevende og utfordrende oppgave. Og det gir sjelden et fullgodt resultat.

I Vansjø i Østfold har man hatt problemer med eutrofiering. Rørslett og Lydersen(1980) gjennomførte en vegetasjonskartlegging som hadde til formål å registrere områder med særlig sterk tilgroing. De påviste at Vanemfjorden/Vestfjorden-bassenget var svært begrodd med helofytter, spesielt av typen sjøsivaks (Scirpus lacustris) og takrør (Phragmites communis).

Helofytter er en gruppe av akvatisk vegetasjon som vokser i strandkanten gjerne i relativt løs substrat. Store deler av aksen stikker opp av vannet, og en populærbetegnelse kan være sivvegetasjon (Mjelde et al. (2008)). Det er denne vegetasjonen som vil være hovedfokus i denne oppgaven.

Jeg har valgt to bilder fra et område ved Feøya. Feøya ligger mellom Vanemfjorden og Vestfjorden. Dette området er godt kartlagt av NIVA både ved feltstudier og ved hjelp av flyfoto. Bildeserien jeg bruker i denne oppgaven er brukt sammen med feltstudier i en rapport av Mjelde et al. (2008). I denne har man brukt en manuell gjennomgang av bildene for å kartlegge utbredelsen av makrofytter i flere områder av Vansjø, deriblant rundt Feøya. I denne fremstillingen vil hovedfokus ligge på den statistiske metoden, så det er ikke forsøkt å trekke noen paralleller til denne rapporten. Det vil bli tema for kapittel 9.

3.1 Refleksjon fra akvatisk vegetasjon

Jordas ulike landskapsformer vil gi forskjellige mønstre i hva som reflekteres av elektromagnetisk stråling. Dette er godt kjent i forhold til det synlige lyset. Det er jo dette som gjør oss i stand til å

oppfatte ulike landskapsformer. Utfordringen er å skille en type vegetasjon fra en annen utelukkende basert på synlig lys. Derfor har man tatt i bruk infrarødt lys som en ekstra parameter. Og dette har vist seg å være en veldig viktig parameter.

Refleksjonen fra vann skiller seg mye fra refleksjonen fra vegetasjon. Ved akvatisk vegetasjon får vi et annet mønster fordi hvert piksel gjerne dekker både vann og vegetasjon. Når vi skal kartlegge akvatisk vegetasjon ønsker vi ikke å telle på individnivå, men se på bestand av en viss utstrekning. Dette innebærer at vi ønsker å inkludere områder hvor det er et stort innslag av vann, dersom det ikke er av stor utstrekning og ligger innimellom områder med vegetasjon.

Det er også forskjeller på akvatisk vegetasjon. Noe har store plantedeler over vannoverflaten, mens andre er delvis under vannoverflaten. Refleksjoner i vannet kan gjøre det vanskelig å observere det som er under vannoverflaten. Dette bør være et kjent tilfelle for de som har ruslet langs vannet og skal se på noe under vannoverflaten. Man må gjerne finne den perfekte vinkelen i forhold til vannoverflaten for å se noe på bunnen. Den samme problemstillingen vil vi oppleve fra et fly eller en satellitt.

Det er gjort utallige undersøkelser på den fysiologiske sammenhengen mellom karakteristikker ved egenskaper i løvet og refleksjonen av elektromagnetisk stråling. Hovedtrekkene er at refleksjonen av synlig lys avhenger av forekomsten og konsentrasjonen av pigmenter. Mens morfologien og innholdet av vann er avgjørende for refleksjonen av infrarødt lys. Mer spesifikt skriver Silva et al. (2007) at nær-infrarødt reflekteres av cellestrukturen, mens midt-infrarødt absorberes av vevsvann. Grønt reflekteres av klorofyllet, mens rødt og blått absorberes av pigmentene.

Valta-Hulkkonen et al. (2003) gjennomførte en studie med en sensor som responderte på tilsvarende frekvensområder som den vi bruker i denne studien. De kom frem til at forskjeller i fargeverdiene mellom ulike arter vel så gjerne kunne være resultat av tettheten på vegetasjonen enn av en egenskap ved artene. Likevel har de kommet frem til en oversikt over hvordan fargeverdiene er for ulike makrofyttarter. Gjennomgående finner vi at refleksjonen fra vegetasjon er høyt korrelert for rødt og grønt.

3.2 Bruksområder for fjernanalyse

Den høye korrelasjonen mellom rødt og grønt gjør at det kan være hensiktsmessig å bytte ut den ene med en variabel som skiller bedre. Behovet for en slik operasjon er et spørsmål om bruksområde for analysen. Dersom bildene skal brukes i en visuell sammenheng er det ofte ønskelig å bruke tre kanaler, fordi digitale bilder ofte er representert med 3 kanaler RGB. Dette vil være typisk for et kartleggingsprosjekt hvor man har en manuell gjennomgang.

I andre sammenhenger er man ikke interessert i å betrakte bildet visuelt, men er mer opptatt av å bruke det i en statistisk modell. Rogan et al. (2001) har en studie hvor de ser på bruk av fjernanalyse i forbindelse med endringer i vegetasjonen. De peker på arbeider som viser at det i en slik sammenheng er en fordel å transformere målingene av fargeverdiene til en indeks. Lehmann et al. (1997) peker på at i sammenheng med observasjoner over en innsjø er normalisert vegetasjonsrefleksjonsindeks (NDVI) og fotokjemisk refleksjonsindeks (PRI) gode indekser for å sammenligne vegetasjon fra ett år til et annet.

Lillesand et al. (1987) gir en utførlig beskrivelse av ulike indekser som brukes i forbindelse med fjernanalyse av vegetasjon. Tasseled cap er betegnelsen på en metode som gir to indekser. Brightness som er summen av alle kanalene, og greeness som er kontrasten mellom infra-rødt og synlige kanaler. Brightness gjenspeiler soil reflectance, mens greenness gjenspeiler mengden av

grønn vegetasjon i bildet. Videre påstås det at dersom man tar høyde for illuminasjon og atmosfæriske effekter kan disse brukes for flere scener.

Jeg har gjort noen små forsøk med bruk av disse indeksene uten at det har ført noen vei. Siden de ikke på noen måte gjør det enklere å tolke bildene visuelt har jeg valgt å ikke fokusere på dem i fremstillingen.

4 Klassifisering

Når man betrakter et flyfoto vil man intuitivt danne seg et bilde av hva det er man ser. Det er mulig å skille mellom skog og vann, og man kan se hvor det er menneskepåvirket. Enten ved jordbruksområder eller med veier og bebyggelse. Det er ulike faktorer ved synsinntrykkene som gir grunnlag for disse vurderingene. Fargene er av stor betydning, og enkelte ganger kan det være mønsteret fargene danner som er avgjørende. En granskog vil danne et annet mønster enn en løvskog.

En annen faktor kan være å se på sammenhenger. Dersom vi ser en vei, som ender i en åpen plass med en rød firkant, er det nærliggende å tenke at det er taket på et hus vi ser. Dette kan være vanskelig å se dersom vi utelukkende ser den røde firkanten av taket. Ser vi noe grønt i kanten av et vann er det nærliggende å tenke at det er siv vi ser. Dersom vi utelukkende ser på de grønne områdene er det vanskelig å tolke dette som områder med siv.

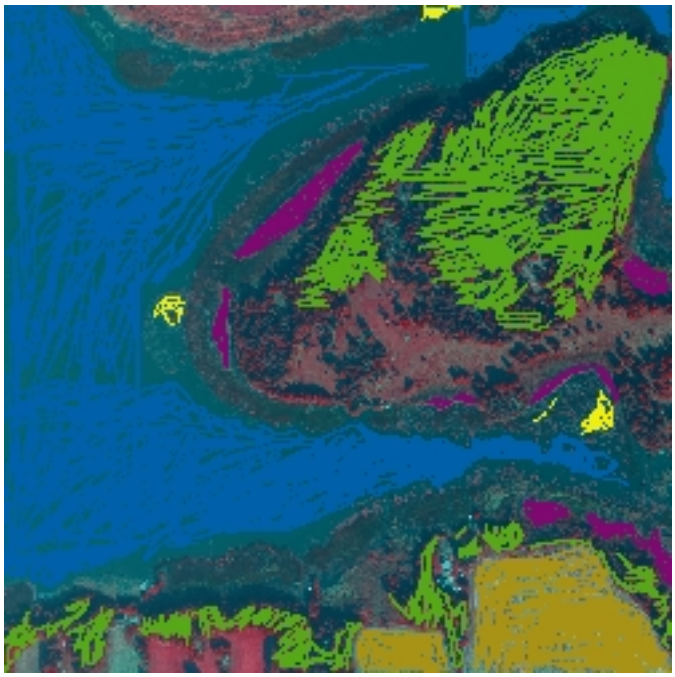
Det er en tidkrevende prosess å tolke et bilde i den hensikt å kartlegge en spesiell vegetasjonstype. I mange tilfeller vil det være ønskelig å ha en automatisert rutine. Dette vil si en rutine som tar for seg hvert enkelt piksel, og avgjør om det tilhører den ene eller andre klassen. En slik rutine vil ofte være avhengig av en viss form for innsats fra brukeren. I sin enkleste form kan dette være at brukeren markerer i bildet enkelte områder han med sikkerhet kan fastslå er av en spesiell klasse. Det er denne fremgangsmåten vi vil forfølge i denne fremstillingen.

4.1 Manuell kartlegging

Utgangspunktet for kartleggingen er en manuell gjennomgang av bildet hvor de mest fremtredende klassene i bildet identifiseres, og markeres med en tykk strek. Hva som skal defineres som en klasse vil som regel være gitt av den oppgaven vi skal løse og hva vi skal kartlegge. Dette vil ikke nødvendigvis være samsvarende med hva som er best fra et statistisk ståsted. Fra et statistisk ståsted vil det være mest formålstjenlig at klassene defineres utfra hvor det er størst variasjon i pikslenes fargeverdier. Dette kan igjen være lite formålstjenlig fra et rent praktisk synspunkt siden klassene man da kommer frem til ikke trenger å gjenspeile vegetasjons- eller landskapstyper.

I denne oppgaven vil hovedfokus være siv-vegetasjon i vannkanten. Som en motsats til siv definerer vi noen andre vegetasjons- og landskapstyper. Dette ble gjort utfra en vurdering av hva jeg ellers så i bildet, og hadde ingen sammenheng med statistiske analyser eller biologiske betraktninger.

Markeringen av siv-vegetasjon ble gjort utfra innspill fra biologen Marit Mjelde. I kapittel 8 vil jeg gå nærmere på betydningen av hvordan disse velges.
I figuren under har jeg lagt inn skravering i ulike farger som gjengir treningssettet som brukes i dette kapittelet og i neste kapittel. I tabellen har jeg lagt inn en definisjon av disse.

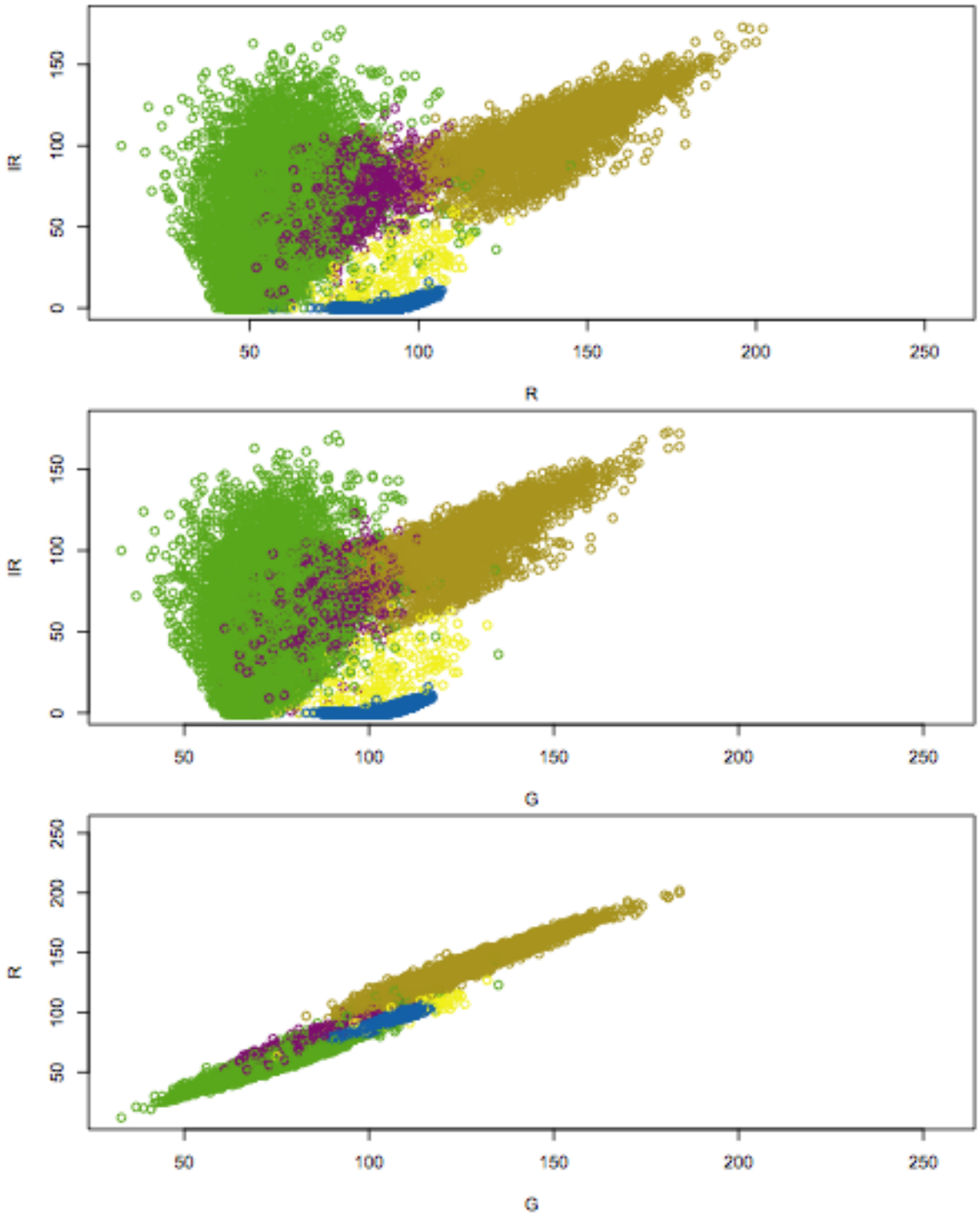


lt	Siv
ønt	Skog
ansje	Jordbruksområder
ått	Vann
la	Myrområder

Figur 3: Bilde som viser treningssettet. Fargene representerer hver sin klasse. Tabellen viser en oversikt over definisjonen av de ulike klassene.

Vårt mål er å kartlegge siv, som er markert med gult i bildet over. Av bildet ser vi at dette utgjør en liten andel av treningssettet. Dette er gjort bevisst siden siv utgjør en liten andel av bildet sett under ett.

I figuren på neste side er det plott som viser spredningen av fargeverdier for de pikslene som utgjør treningssettet for de fem klassene.



Figur 4: Spredning i fargeverdi for pikslene som utgjør de ulike treningssettene. Øverst er en sammenstilling mellom infra-rødt og rødt. I midten ser vi infra-rødt og grønt, mens nederst har vi rødt mot grønt. Infra-rødt har verdier i intervallet 0-150, mens rødt og grønt har verdier i intervallet 50-250.

Av plottene ser vi at det er en høy korrelasjon mellom rødt og grønt. Det ser vi av at de to øverste grafene er veldig like. Vi ser det også i den nederste hvor punktene legger seg på en rett linje. Ellers ser vi at klassene danner ellipseformede klynger.

Vår hovedinteresse siv er markert med gult, og det kan virke som at denne klassen skiller seg litt ut fra de øvrige. I den nederste grafen ser vi at det er ganske mye overlapp med blått som representerer vann.

4.2 Multivariabel normalfordeling

Hvert piksel kan betegnes som et objekt. Til objektet er det knyttet et sett med variabler y_i og en ukjent størrelse x_i. Variablene y_i vil i utgangspunktet være de observerte fargeverdiene i hvert piksel, men her kunne vi også lagt inn en av indeksene fra kapittel 3.1. Som nevnt i kapittel 1.2 vil jeg bruke en enkelt indeks i for hvert piksel. Implisitt i denne ligger en overgang til det 2-dimensjonale rutenettet i bildet.

Størrelsen x_i representerer klassetilhørigheten. Denne kan anta distinkte verdier $k \in 1, 2, ..., K$. Vektoren $x^* = \{x_1, x_2, ..., x_n\}$ representerer den sanne klassetilhørighet, og er vårt ønskede resultat.

En vanlige tilnærming til denne problemstillingen er en sannsynlighetsmodell hvor man antar at objektene i hver klasse antar en multivariabel normalfordeling som er avhengig av klassetilhørigheten

$$f_k(Y_i = y_i | x_i = k) \sim N(\mu_k, \Sigma_k).$$

- y_i er vektor med observerte verdier i pikselet
- x_i er klassetilordning for pikselet
- μ_k vektor med klassens middelverdier
- Σ_k klassens kovariansmatrise

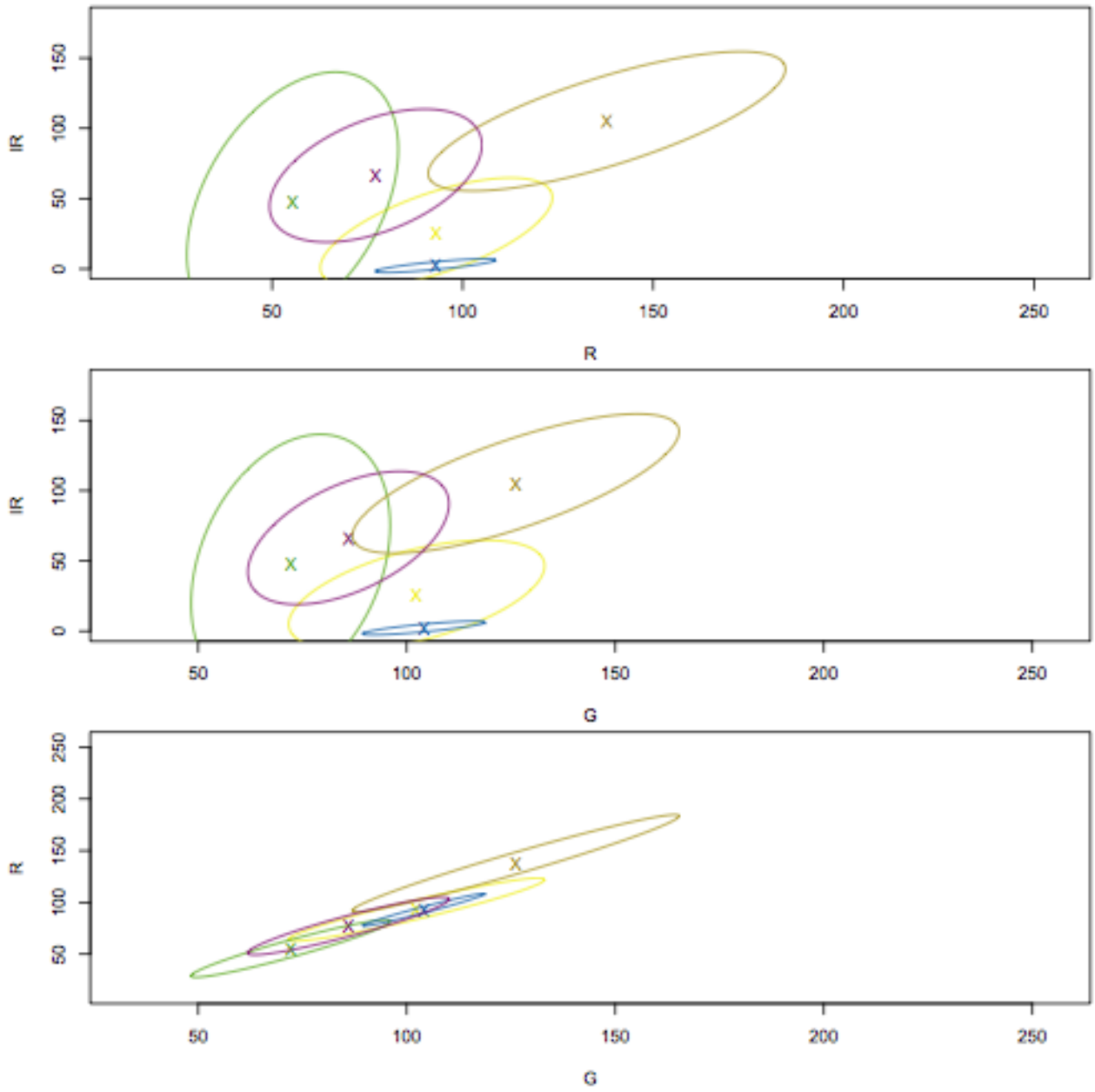
For å estimere parameterene i de multivariable normalfordelingene kan vi benytte data fra treningssettet og følgende formler:

$$\hat{\mu_k} = \frac{1}{N_k} \sum_{i=1}^{N_k} y_i$$

$$\hat{\Sigma_k} = \frac{1}{N_k - 1} \sum_{i=1}^{N_k} (y_i - \hat{\mu_k})(y_i - \hat{\mu_k})^T$$

- N_k antall piksler i treningssettet fra klassen k

I figuren under er de 5 fordelingene presentert med ellipser som er trukket opp for 95 % sannsynlighetsmasse. Variablene IR, R, G er satt opp mot hverandre i 3 grafer.



Figur 5: Middelverdi (X) og 95 % sannsynlighetsmasse for de ulike klassene.

Dersom vi setter denne figuren opp mot figur 4, ser vi at antakelsen om normalfordeling er troverdig. Vi ser også at det er mye overlapp mellom klassene.

4.3 Maksimal sannsynlighet

Forventningsverdien for de ulike klassene er ikke tilstrekkelig for å fastslå om et piksel skal tilhøre den ene eller den andre klassen. Vi må se på sannsynligheten for at pikselet tilhører en klasse k gitt fargeverdien y_i. Dette er en klassisk Bayes klassifisering hvor vi skal finne den x_i som maksimerer

$$P(X_i = x_i | Y_i = y_i).$$

Med Bayes teorem blir dette

$$P(X_i = x_i | Y_i = y_i) = \frac{\pi_{x_i} f_{x_i}(y_i)}{\sum_{k=1}^{K} \pi_k f_k(y_i)}$$

Her er π en relativ størrelse som angir fordelingen mellom klassene over hele bildet. Dette er under antakelsen om at bildet kun inneholder de K klassene vi har markert, slik at

$$\sum_{k=1}^{K} \pi_k = 1\,.$$

4.4 Estimering av π

En enkel tilnærming er å bruke estimatene for $\{\mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k\}$ direkte fra treningssettet, og estimere den relative fordelingen $\{\pi_1, ..., \pi_k\}$ gjennom en iterativ løkke. Et utgangspunkt for estimatet kan være fordelingen mellom klassene i treningssettet

$$\pi_k = \frac{N_k}{N_T}.$$

- N_k - Antall piksler i klasse k i treningssettet
- N_T - Totalt antall piksler i treningssett

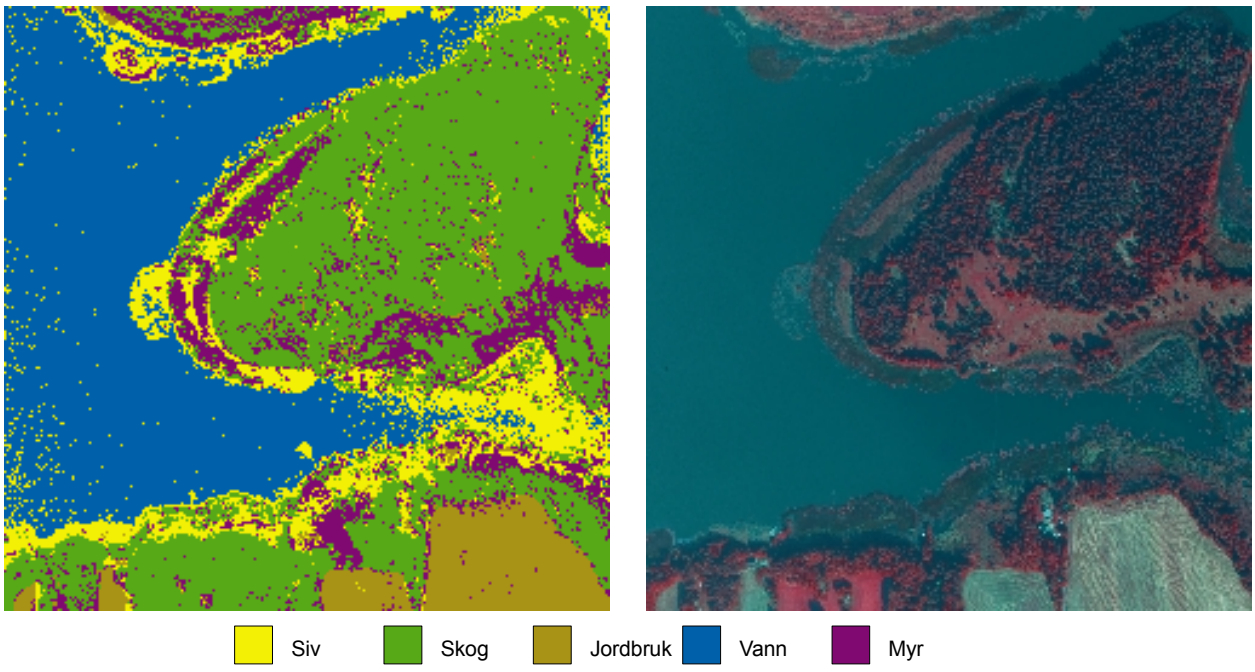
Fordelingen i treningssettet trenger ikke nødvendigvis ha betydning for fordelingen i bildet i sin helhet. Her kan vi også velge å bruke

$$\pi_k = \frac{1}{K}.$$

Vi gjennomfører en klassifisering basert på disse parameterene. Utfallet av denne klassifiseringen brukes deretter som grunnlag for et nytt estimat av $\{\pi_1, ..., \pi_k\}$. Denne gang som fordelingen over hele bildet. Dette gjentar vi inntil parameterene konvergerer. Våre forsøk viser at dette skjer etter ganske få iterasjoner. Dette fremkommer av tabellen under.

Siv	Skog	Jordbruk	Vann	Myr
0,007	0,285	0,101	0,564	0,043
0,060	0,398	0,072	0,363	0,107
0,111	0,368	0,067	0,335	0,120
0,138	0,356	0,066	0,318	0,123
0,150	0,351	0,066	0,309	0,123
0,156	0,350	0,066	0,306	0,123
0,158	0,350	0,066	0,304	0,123
0,159	0,350	0,066	0,303	0,123
0,159	0,349	0,066	0,303	0,123
0,160	0,349	0,066	0,303	0,123

Tabell 1: Utvikling av parameteren π for hver klasse etter hver iterasjon.



Figur 6: Klassifiseringsresultat vises t.v., og originalbildet t.h. Vi ser at hovedtrekkene er riktige, men at siv blir altfor omfattende. I tillegg er resultatet preget av mye støy.

Figuren over viser klassifiseringsresultatet etter at vi har tilpasset π. Her ser vi at vi klarer å gjenskape hovedtrekkene i bildet, men at vi får en overrepresentasjon av siv. Nærmere studier av bildet viser at det ikke er enkeltflekker flytende ute i vannet slik resultatet skulle tilsi. Også inne i vika under Feøya er det for mye siv. Generelt kan vi si at det er for mye støy i bildet.

I kapittel 4.6 vil jeg komme tilbake til andre metoder for å estimere parameterene, men først vil jeg se på hvordan vi kan kvantifisere feilene i klassifiseringen.

4.5 Forvirringsmatrise

For hvor god er egentlig klassifiseringen? Over prøvde jeg meg med noen subjektive betraktninger, men for å belyse dette ordentlig bør vi ha noen tall å vise til. En måte å gjøre det på er gjennom å sette opp en forvirringsmatrise. Dette er en tabell som viser antallet piksler i de ulike klassene i et testsett og i utfallet av klassifiseringen. Det finnes ulike måter å sette opp en forvirringsmatrise på. Jeg har valgt å benytte tilsvarende som Banko (1998) skisserer. I kapittel 4 og 5 vil jeg bruke treningssettet som testsett.

For hver rad har vi antallet som ble klassifisert i en gitt klasse. Disse er fordelt på hver sin kolonne avhengig av hvilken klasse de var knyttet til i treningssettet. Dermed får vi at antallet i diagonalen uttrykker hvor mange av pikslene som er klassifisert i den klassen de hadde i treningssettet. Tallene utenom diagonalen uttrykker feil i klassifiseringen.

Feilene kan tolkes på to måter når vi fokuserer på en klasse A. Vi har antallet som havner i klasse A, men som egentlig var i klasse B. Dersom dette antallet er høyt kan vi si at modellen har dårlig presisjon for A. Den andre feilen er de som opprinnelig var i klasse A, men som havner i klasse B. Dersom dette antallet er høyt kan vi si at modellen har dårlig sensitivitet for A.

I forvirringsmatrisen har vi summert antallet som klassifiseres i en klasse i kolonnen Sum. Tilsvarende har vi en rad Sum for summen av piksler i hver klasse i treningssettet. Presisjon finner vi ved å dele antallet i diagonalen på summen i kolonnen, og uttrykke det som en prosentandel. Sensitivitet finner vi ved å dele antallet i diagonalen med summen i raden sum, og uttrykke det som en prosentandel.

Til slutt har vi en total på enden av diagonalen. Dette er antallet piksler i treningssettet. Og vi har en total nøyaktighet. Dette er summen av piksler som klassifiseres korrekt dividert med totalt antall piksler uttrykt som en prosentandel.

		Treningssett						
		Siv	Skog	Jordbruk	Vann	Myr	Sum	Pres.
Klassifisert	Siv	183	48	4	1332	18	1585	11.55%
	Skog	2	6905	1	4	236	7148	96.60%
	Jordbruk	0	17	2565	0	15	2597	98.77%
	Vann	2	0	0	13174	0	13176	99.98%
	Myr	2	354	42	0	836	1234	67.75%
	Sum	189	7324	2612	14510	1105	25740	
	Sens.	96.83%	94.28%	98.20%	90.79%	75.66%		
						Treffpros.	91.93%	

Figur 7: Forvirringsmatrise etter klassifisering med tilpasning av π. På hver rad vises antallet piksler som er klassifisert til en klasse. Disse er fordelt i kolonner avhengig av hvilken klasse pikselet var knyttet til i treningssettet. Dermed utgjør diagonalen antallet piksler som klassifiseres korrekt. I kolonnen Pres. (presisjon) er dette antallet dividert med summen av piksler som klassifiseres i den klassen, mens i raden Sens. (sensitivitet) er antallet dividert med summen av piksler som var knyttet til klassen i treningssettet. Treffpros. er summen av antall korrekte klassifiseringer dividert med totalt antall piksler.

Av tabellen ser vi at vi har dårlig presisjon på siv. Dette skyldes at mange av de pikslene vi klassifiserer som siv var vann i treningssettet. Tilsvarende ser vi at mye av det som klassifiseres som myr var skog i treningssettet. Klassen med best presisjon er vann, mens klassen med best sensitivitet er jordbruksklassen. Dette gjenspeiler mye av det vi så i bildet.

Forvirringsmatrisen kan være et greit verktøy for å å få en oversikt over hva som går feil og hva som fungerer bedre i modellen vår, men den er lite praktisk utover det. Det er vanskelig å bruke tallene i forvirringsmatrisen som et mål på feil, og justere modellen utfra det. Det ville vært enklere dersom vi kunne fremskaffe ett enkelt tall, som forteller oss hvor god modellen er. Feilprosenten kan være ett slikt mål. Denne er forholdet mellom antallet korrekte klassifiseringer og totalt antall piksler i testsettet.

Siden vårt hovedfokus er å kartlegge siv vil vi lage en koeffisient med fokus på feilprosent i klassifisering av siv. Først definerer vi noen størrelser slik de er i Hastie et al. (2001) kapittel 9:

$$S_S = \frac{N_d}{N_t}\text{- Sensitivitet for siv}$$

$$S_P = \frac{N_d}{N_k}\text{ - Presisjon for siv}$$

$$S_F = \frac{N-N_d}{N-N_t}\text{ - Spesifisitet for siv}$$

- N_d - Antall piksler som er siv i treningssettet og som klassifiseres som siv.
- N_t - Antall piksler som er siv i treningssettet
- N_k - Antall piksler som klassifiseres som siv
- N- Antall piksler totalt

Sensitivitet er som tidligere beskrevet et mål på hvor god modellen vår er til å fange opp alle piksler

med siv. Vi tar utgangspunkt i de pikslene som er siv i treningssettet og ser på andelen som også blir klassifisert som siv. En modell som klassifiserer alle pikslene i bildet som siv vil ergo ha en meget god sensitivitet.
Dette sier oss at vi ikke bare kan legge vekt på sensitiviteten. Vi må også se på hvor god modellen er til å unngå piksler av andre klasser. Dette fremkommer med presisjonen. Dette er et mål på hvor god modellen er til utelukkende å trekke ut de pikslene som var siv i treningssettet.
Spesifisitet er et mål på hvor god modellen er til å klassifisere som noe annet, det som ikke var siv i treningssettet. Av formelen ser vi at dette måltallet tilsvarer sensitiviteten, men at det tar høyde for forholdet mellom antall piklser i klassen siv og antall piksler totalt. Dermed vil dette måltallet ha større betydning dersom man skal sammenligne en modell på ulike testsett enn når vi skal sammenligne flere modeller på samme sett.
Middelet mellom sensitivitet og presisjon gir en god indikasjon på hvor god modellen er til å klassifisere siv

$$S = \frac{S_S + S_P}{2}.$$

Det er også mulig å legge inn en vekting av de to størrelsene. Dersom det er viktig å få med seg alle mulige forekomster av siv i vassdraget vekter man sensitivitet høyt. Dersom det er viktigere at man skal kunne stole på det resultatet man får direkte, kan man vekte presisjonen høyere.

Det viste seg raskt at dette måltallet egnet seg dårlig i situasjoner hvor ingen eller få piksler ble klassifisert som siv. Dersom ingen piksler blir klassifisert som siv vil S_P være udefinert, og dersom ett piksel blir korrekt klassifisert som siv vil vi få en $S_P = 1$. For å unngå dette brukte jeg et alternativt formelsett

$$S_P = \frac{N_d}{N_k + 1}.$$

Tilsvarende ble gjort for sensitivitet for å opprettholde konsistens mellom de to koeffisientene

$$S_S = \frac{N_d}{N_t + 1}.$$

4.6 Tilpasning av parametere ved bruk av EM

I kapittel 4.4 brukte vi en enkel tilnærming til løsning av klassifiseringen. Denne la stor vekt på treningssettet. En mer avansert tilnærming er Estimation Maximization (EM). Denne forsøker å ta hensyn til hele bildet når den estimerer parametersettet:

$$\phi = \{\pi_1, ..., \pi_k, \mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k\} .$$

Det den skal komme frem til er den ϕ som maksimerer $P(X = x|Y = y)$.
EM-algoritmen brukes gjerne i situasjoner hvor man skal maksimere sannsynligheten i en modell hvor det er vanskelig å finne et direkte uttrykk for sannsynligheten. Tanken er at man utvider treningssettet med ikke-observerte data.

De initielle verdiene for ϕ henter vi fra treningssettet som tidligere. Dette bruker vi til å klassifisere bildet, og utfallet danner grunnlag for estimering av et nytt parametersett. I denne estimeringen bruker vi et sett med vekter for hver piksel

$$w_{i,k} = P(X_i = k|y_i, \phi_k).$$

Denne vektingen innebærer at vi vil beholde en viss innflytelse fra treningssettet.

De nye estimatene blir:

$$\hat{\mu_k} = \frac{1}{W_k} \sum w_i y_i$$

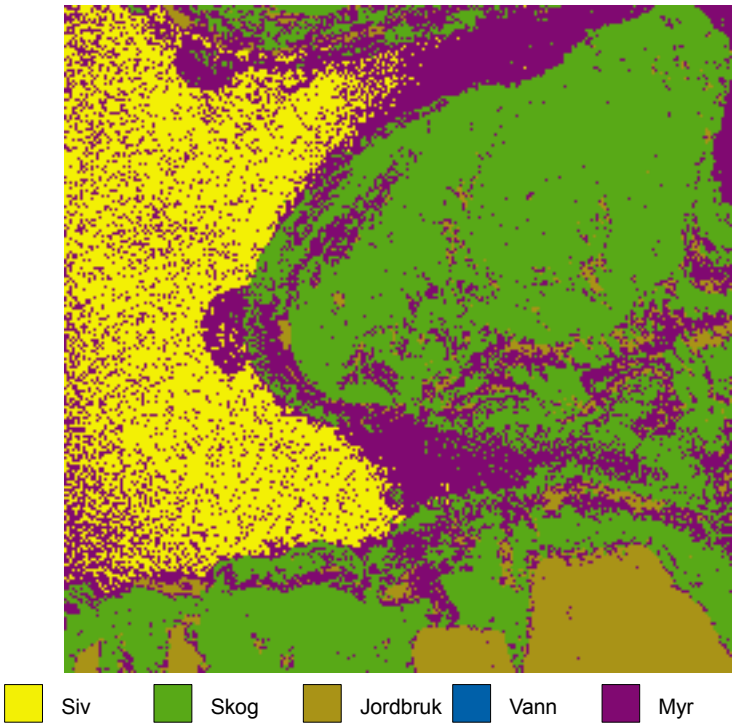
$$\hat{\Sigma_k} = \frac{1}{W_k} \sum w_k (y_i - \hat{\mu_k})^T (y_i - \hat{\mu_k})$$

$$\hat{\pi_k} = \sum w_i.$$

Her er W_k en normaliseringsfaktor gitt av

$$W_k = \sum w_{i,k}.$$

Prosessen gjentas inntil man oppnår en stabil klassifisering. I vårt tilfelle synes dette vanskelig å oppnå. I bildet på neste side ser vi at klassen vann har blitt spist opp av siv og de opprinnelige områdene med siv er klassifisert som myr. Dette skyldes at vektingen vår ikke favoriserer opprettholdelsen av 5 klasser. Dette mønsteret kan vi lese ut av forvirringsmatrisen på neste side.



Figur 8: Resultat av full EM. Vi ser at vann er fraværende. Mens siv og myr har fått et ufortjent stort omfang.

		Treningssett						
		Siv	Skog	Jordbruk	Vann	Myr	Sum	Pres.
Klassifisert	Siv	1	0	0	9429	0	9430	0.01%
	Skog	20	7088	2	310	493	7913	89.57%
	Jordbruk	14	40	2591	0	86	2731	94.87%
	Vann	0	0	0	0	0	0	NaN%
	Myr	154	196	19	4771	526	5666	9.28%
	Sum	189	7324	2612	14510	1105	25740	
	Sens.	0.53%	96.78%	99.20%	0.00%	47.60%		
						Treffpros.	39.65%	

Figur 9: Forvirringsmatrise etter full EM.Vi ser at siv har forflyttet seg helt til de fargeverdiene som utgjorde vann, mens myr sprer seg utover alle klasser i treningssettet. Betegnelsen NaN% for vann skyldes at det ikke finnes noen observasjoner i klassifiseringsresultatet.

4.7 Justert kovariansmatrise

I kapittel 4.5 så vi at vi hadde problemer med å skille siv og vann. Dette skyldes at forvetningsfunksjonene for disse to klassene er overlappende. Hjort (1986) så på feilraten man må forvente dersom man har to eller flere klasser med multivariable normalfordelinger, og gjennomfører en klassifisering.

Utgangspunktet er Mahalanobis distansen

$$\delta = \{(\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)\}^{1/2}.$$

I situasjoner hvor vi ikke har felles kovariansmatrise kan denne tilnærmes med

$$\Sigma = \frac{1}{2}(\Sigma_1 + \Sigma_2).$$

Hjort foreslo en generalisert Mahalanobis distanse

$$\omega = (\delta^2 + \gamma^2)^{1/2}$$

hvor

$$\gamma^2 = 4log\frac{|\Sigma|}{|\Sigma_1|^{1/2}|\Sigma_2|^{1/2}}.$$

Verdien ω bør oversige 2,5 for å få en feilrate under 10%.

	Siv	Skog	Jordbruk	Vann	Myr
Siv	0	4,4	5,6	2,8	3,9
Skog	4,4	0	6,7	5,7	2,4
Jordbruk	5,6	6,7	0	9,9	4,6
Vann	2,8	5,7	9,9	0	7,1
Myr	3,9	2,4	4,6	7,1	0

Tabell 2: Generalisert Mahalanobis differanse mellom klassene gitt at middel og kovarians estimeres direkte fra treningssettet. Siden vi bruker datasett med 3 dimensjoner vil verdier over 2,5 svare til en feilrate lavere enn 10%.

Av tabellen ser vi at det er skillet mellom siv/vann og skog/myr, som ser ut til å være en utfordring.

Forvirringsmatrisen i figur 7 antyder at dette stemmer godt. Klassen siv får mange piksler som kommer fra treningssettet vann, og klassen skog får mange av de som var i klassen myr.

Dette gir en idé om at man kan justere litt på kovariansmatrisen for å skille bedre mellom klassene. Her finnes det flere tilnærmelser. Friedman (1989) introduserte en klassifiseringsmodell hvor man påvirker hver enkelt klasses kovariansmatrise med den som er felles for hele treningssettet. Hans motiv for å gjøre det var i tilfeller med begrensede treningssett. I disse situasjonene kan enkeltobservasjoner ha stor betydning for kovariansmatrisen. Ved å trekke inn en felles kovariansmatrise ønsket han å unngå denne effekten.
Celeux og Govaert (1993) presenterte en annen modell med samme formål. De tok utgangspunkt i en eigenvalue dekomposisjon av kovariansmatrisene for klassene

$$\Sigma_k = \lambda_k D_k A_k D_k^T.$$

Ved å gjøre λ_k, D_k og A_k felles for alle klassene kan man påvirke betydningen av de ulike treningssettene. Matrisene D og A kan begrenses ytterligere ved å slå dem sammen til en felles global matrise, og man kan sette $\lambda = 1$. Til sammen kom Celeux og Govaert 14 modeller for påvirknig av kovariansematrisen i en modell med multivariable normalfordelinger.

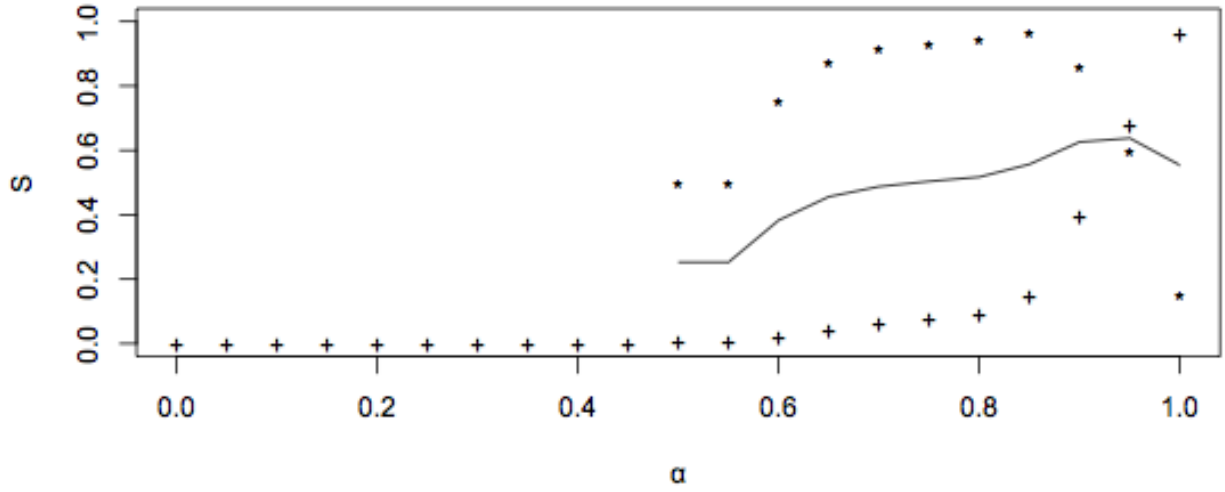
Til tross for at det i mitt tilfelle ikke er problemer med et begrenset treningssett, men overlapp mellom klassene, virker Friedmans tilnærming å fungere best av disse to.

En forenkling av Friedmans modell kan uttrykkes med:

$$\Sigma_k(\alpha) = \alpha\Sigma_k + (1-\alpha)\Sigma$$

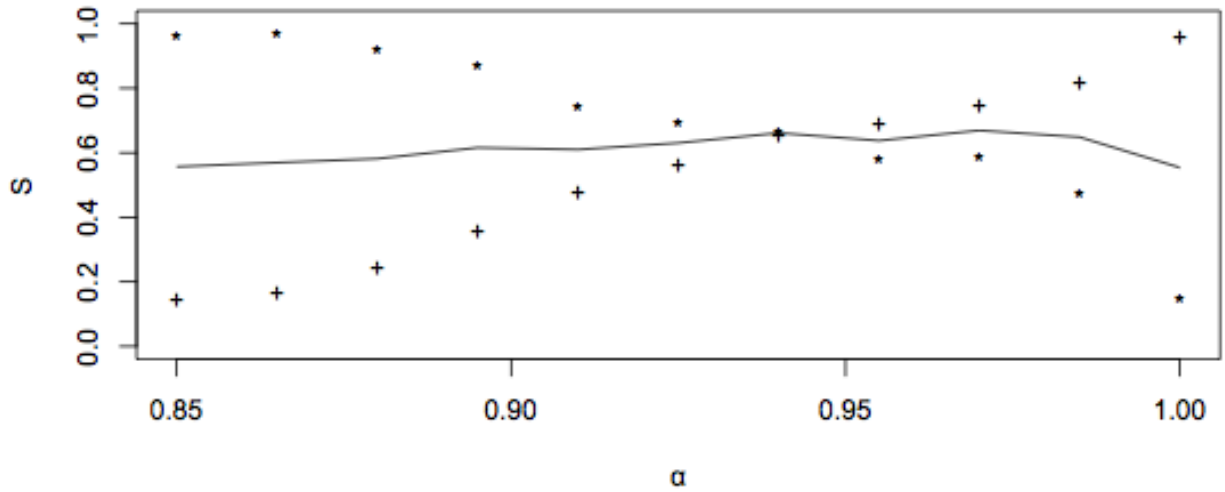
$$\alpha \in [0, 1].$$

Her er Σ_k som tidligere hver gruppes kovarians, mens Σ er den universelle kovariansen.
Nå er det π_k og α vi ønsker å estimere, mens μ_k, Σ_k og Σ beholder sine verdier fra treningssettet. Vi finner den optimale α ved å kjøre en full klassifisering for ulike α og deretter beregne koeffisienten S fra kapittel 4.5.



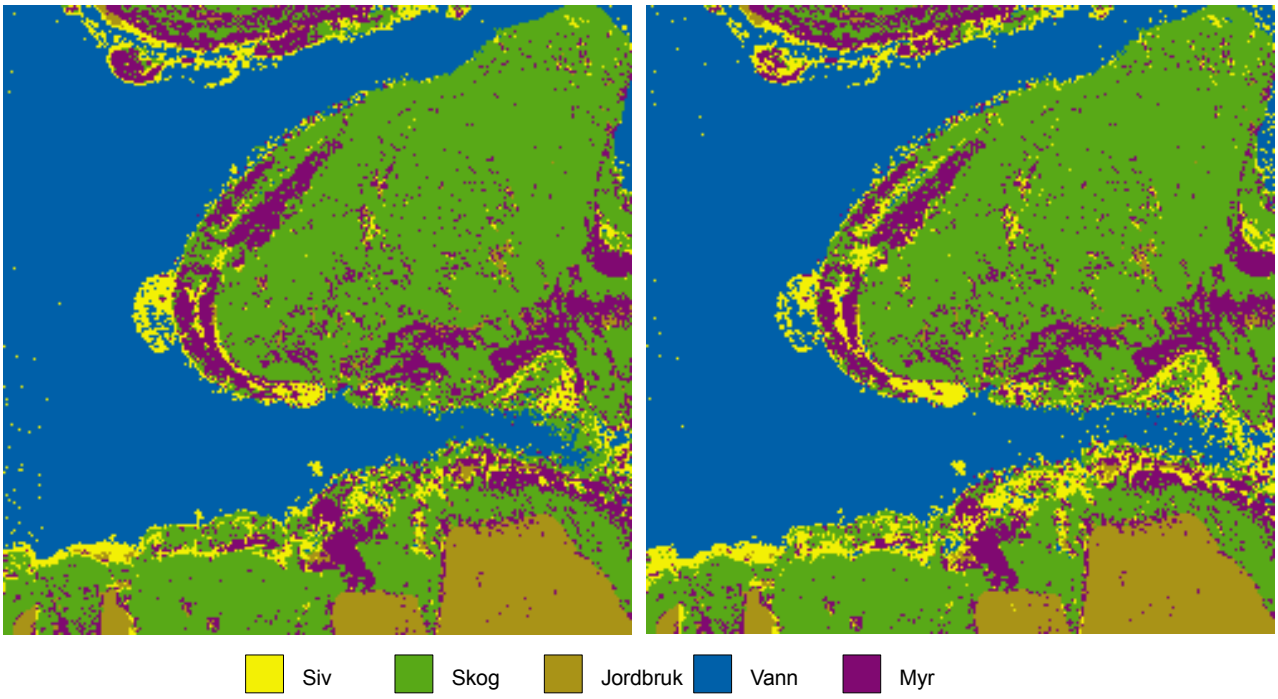
Figur 10: Presisjon (*) og sensitivitet (+) for siv gitt ulike α. Den heltrukne linjen viser S.

I figuren viser den heltrukne streken utviklingen av S for varierende α. I intervallet 0 til 1 er den økt med 0,05 for hvert steg. Sensitivitet er markert med *, mens presisjon er markert med +.
Det vi ser er at verdiene er udefinert frem til 0,5. Dette skyldes at ved lavere verdier blir ingen piksler klassifisert som siv. Deretter har vi en høy presisjon, mens sensitiviteten øker gradvis. Dette skyldes at en økende α tilsier at vi spisser mer og mer mot parameterene som skiller siv fra de øvrige. Ved α=0,85 har vi maksimal presisjon. Deretter avtar denne. Noe som skyldes at vi etter denne verdien trekker til oss andre piksler fra andre klasser. Samtidig ser vi at sensitiviteten øker. Her er vi altså i et område hvor grensene mellom de ulike klassene er vanskelig å definere.



Figur 11: Tilsvarende som forrige figur, men denne gangen med langt finere inndeling av α.Maksimal S fremkommer ved α=0,985.

Den siste grafen på forrige side viser resultatet av å kjøre klassifiseringen med enda finere inndeling. Her varierte α i intervallet 0,85 til 1 med et intervall på 0,15. Maksimal S fremkommer ved α=0,985.



Figur 12: Resultat av klassifisering med regularisert kovariansmatrise t.v., og uten t.h.

I figuren over vises to bilder basert på klassifisering med bruk av regularisert kovariansmatrise til høyre og uten til venstre. Den største forskjellen er at en del av de frittstående gule pikslene i venstre kant har forsvunnet. På samme måte har det gule «øret», på venstre side av halvøya midt i bildet, blitt spist opp av blått. Dette viser at endel av det gule har gått over til å bli blått. På den annen side er det mulig å se at endel av de gule sonene i randsonen til vannet har blitt større. Dette har skjedd på bekostning av grønt og lilla.

Dette kan vi studere nærmere på neste side ved å se på utviklingen av forvirringsmatrisen ved ulike α.

		Treningssett					
	Siv	Skog	Jordbruk	Vann	Myr	Sum	Pres.
Klassifisert Siv	28	1	0	0	0	29	96.55%
Skog	9	7051	2	3	366	7431	94.89%
Jordbruk	0	15	2541	0	7	2563	99.14%
Vann	104	6	0	14507	4	14621	99.22%
Myr	48	251	69	0	728	1096	66.42%
Sum	189	7324	2612	14510	1105	25740	
Sens.	14.81%	96.27%	97.28%	99.98%	65.88%		
					Treffpros.	96.56%	

Figur 13: Forvirringsmatrise etter klassifisering med α=0,85

		Treningssett					
	Siv	Skog	Jordbruk	Vann	Myr	Sum	Pres.
Klassifisert Siv	155	37	2	118	13	325	47.69%
Skog	4	6960	1	6	269	7240	96.13%
Jordbruk	0	18	2567	0	15	2600	98.73%
Vann	25	0	0	14386	0	14411	99.83%
Myr	5	309	42	0	808	1164	69.42%
Sum	189	7324	2612	14510	1105	25740	
Sens.	82.01%	95.03%	98.28%	99.15%	73.12%		
					Treffpros.	96.64%	

Figur 14: Forvirringsmatrise etter klassifisering med α=0,985.

		Treningssett					
	Siv	Skog	Jordbruk	Vann	Myr	Sum	Pres.
Klassifisert Siv	181	44	3	963	15	1206	15.01%
Skog	2	6949	1	5	261	7218	96.27%
Jordbruk	0	18	2567	0	15	2600	98.73%
Vann	2	0	0	13542	0	13544	99.99%
Myr	4	313	41	0	814	1172	69.45%
Sum	189	7324	2612	14510	1105	25740	
Sens.	95.77%	94.88%	98.28%	93.33%	73.67%		
					Treffpros.	93.45%	

Figur 15: Forvirringsmatrise etter klassifisering med α=1,0.

Ved α=0,85 har vi en høy presisjon, men en lav sensitivitet. D.v.s. at de pikslene vi klassifiserer som siv også var siv i treningssettet. Faktisk har vi bare èn bom. Dette var skog i treningssettet. Denne presisjonen kommer fordi vi har en veldig dårlig sensitivitet. Modellen vår fanger opp få, faktisk bare 15 %, av de pikslene som ble klassifisert som siv.
Dette bedrer seg ved α=0,985. Nå har vi fått med 82% av pikslene som var siv i treningssettet. Dette går på bekostning av presisjonen, som nå er nede i 48%. I hovedsak er dette piksler som var vann eller skog i treningssettet. Dersom vi bruker α=1,0 får vi enda bedre sensitivitet men desto dårligere presisjon. Nå ser vi også at den totale treffprosenten går ned.

Det vi ser er at klassifiseringen mellom siv og vann er den store utfordringen. Dette er ingen stor overraskelse siden vi er klar over at når vi kartlegger siv, så vil vi få med oss piksler som i sin karakter er vann. Dette skyldes at det i realiteten er mest vann akkurat i det området denne pikselen dekker, men i forbindelse med en kartlegging av området vil det for oss være mer aktuelt å fremstille denne lokaliteten som siv. Dette er ikke i egenskap av sin fargeverdi, men mer i egenskap av sin nærliggenhet til andre piksler med mer fremtredende karakteristikker. Her snakker vi om pikselets naboskap eller konteksten, og dette er et forhold jeg vil se nærmere på i neste kapittel.

5 Kontekstuell klassifisering

I forrige kapittel så vi på kartlegging av siv ved å se på fargeverdien i enkeltstående piksler. Metoden vi brukte for kartleggingen baserte seg på en klassifisering gitt sannsynlighetsfordelingen til utvalgte klasser. Klassene var bestemt fra en manuell gjennomgang av bildet. Foruten siv som er vårt hovedfokus hadde vi valgt ut klasser som syntes å være relevante.

Resultatet av klassifiseringen viste at 96% av treningssettet vårt ble klassifisert i sin opprinnelige klasse. Dette antyder at metoden vår er god, men ser vi på bildene i figur 12 er ikke resultatet like overbevisende. Det er mange små flekker av siv på områder hvor det er lite trolig å finne siv. I tillegg er avgrensningen mellom klassene lite markerte. Dette kan være ønskelig dersom vi skal bruke metoden i forbindelse med kartlegging. Slik resultatet fremstår nå er det vanskelig å trekke en klar grense mellom det som inngår i klassen siv og det som er i klassen vann.

Som vi så i kapittel 3 kan dette skyldes at sivet vokser i vann og at refleksjonen i disse pikslene er en sammenblanding av refleksjon fra sivaksene og vannoverflaten. Dette vil gjenspeiles i fargeverdien. Et annet forhold er at sensoren kan ha en høyere oppløsning enn hva som er ønskelig for vår kartleggingsoppgave. Flybildene vi benytter i denne oppgaven har en oppløsning på 20*20 cm. Dette er en oppløsning som overgår den ønskede oppløsningen i kartleggingen.

Når vi markerer områder som siv i bildet vil enkeltpiksler nødvendigvis representere vann. Siden det er naturlig å finne vann innimellom områdene med siv. I det endelige resultatet er ikke disse enkeltpikslene interessante. For det endelige produktet er det ønskelig at dette fremstår som et enhetlig område av siv.

Dermed kan det synes litt unødvendig å øke oppløsningen i bildet. Kunne man ikke like godt benyttet seg av bildene fra en satellitt med oppløsning på 15*15 m. Fordelen er at vi kan styre hvordan informasjonen som ligger i et sett med piksler skal sammenstilles til et felles resultat. I tilfeller med lav oppløsning, f.eks. fra en satellitt, skjer denne sammenstillingen i sensoren og vi går glipp av viktig informasjon. Når vi har informasjonen fra hvert av disse pikslene kan vi selv bestemme metoden for å sammenstille til ett signal.

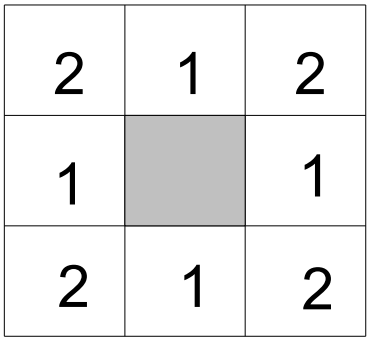
5.1 Markov random field

En av disse metodene er å anta at sannsynligheten for at et piksel tilhører en klasse er gitt av hvilke

klasser naboene tilhører. Dette kan uttrykkes gjennom en Markov random field (MRF)

$$P(X_i = x_i | x_j, j \in N_i).$$

Her uttrykker N_i et sett av piksler som danner naboskap med piksel i. Definisjonen av et naboskap kan variere fra modell til modell. I forbindelse med rektangulære rutenett er det vanlig å omtale et naboskap som første eller andre ordens. Et første ordens naboskap tar kun hensyn til de 4 rutene som ligger kant i kant med det aktuelle pikselet. I figuren under er disse markert med 1. Et andre ordens naboskap vil i tillegg se på hjørnerutene merket med 2.



Figur 16: Piksler som inngår i naboskapet til midten. Første ordens naboskap (1) og andre ordens naboskap (1+2).

Et annet sentralt moment i forhold til naboskap er klikker. En klikk består av piksler som er gjensidige naboer. I tillegg betraktes enkeltstående piksler som en klikk.

En MRF ser på naboskapet rundt ett enkelt piksel. Vi er ute etter en modell som tar for seg hele bildet, og definerer en global naboskapsmodell. Denne koblingen får vi gjennom Hammersley-Cliffords teorem. Besag (1974) gir et bevis for dette teoremet. I dette ligger følgende vilkår. Det må være et begrenset sett med klasser K, og

$$P(X = x) > 0 \text{ for alle x.}$$

Dersom disse vilkårene er oppfylt vil vi ha

$$P(X_i = x_i | x_j, j \in N_i) = P(X_i = x_i | x_j, j \neq i).$$

Av dette følger at en lokal MRF, også kan ses på som en global sannsynlighetsmodell for klassetilhørigheten. Den globale modellen har samme naboforhold som den lokale.

En global sannsynlighetsmodell er Gibbs sannsynlighetsmodell (GRF)

$$P(X = x) = \frac{e^{-U(x)}}{Z},$$

hvor $U(x)$ er en energifunksjon og Z en normaliseringskonstant. $U(x)$ kan uttrykkes ved hjelp av klikkene

$$U(x) = -\sum_{q \in Q} V_q(x_q)\,.$$

Her er V_q en klikkfunksjon, som kan varieres i forhold til klikkens utforming. Vi kan ha en funksjon for enkeltstående klikker og en annen for parvise klikker. For parvise klikker kan vi også velge å ha ulike funksjoner for de horisontale og de vertikale.

5.2 Kontekstuell sannsynlighetsfunksjon

Besag (1986) foreslo følgende modell for den kontekstuelle sannsynligheten i forbindelse med klassifisering av bilder

$$P(X_i = x_i | x_j, j \in N_i) = \frac{e^{\beta H_{N_i}(x_i)}}{\sum_{k=1}^{K} e^{\beta H_{N_i}(k)}}.$$

Her er β en parameter som uttrykker graden av kontekst. Funksjonen

$$H_{N_i}(k)$$

uttrykker hvor mange av naboene i N_i som er av klassen k, og tilsvarer klikkfunksjonen vi beskrev i forrige avsnitt. I dette tilfellet gjelder den kun for parvise klikker, men tar ikke hensyn til om de er horisontale, vertikale eller diagonale. Dette innebærer at klikkfunksjonen for enkeltstående klikker i dette tilfellet er lik 0.

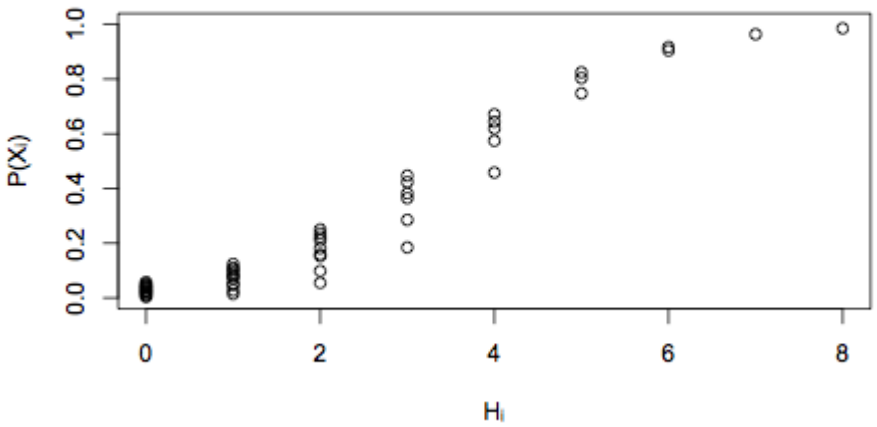
Når vi bruker en andre ordens MRF, og ser bort fra kantene, kan denne funksjonen ha verdier fra [0,..8]. Samtidig vil vi ha summen

$$H_{N_i}(1) + H_{N_i}(2) + ... + H_{N_i}(K) = 8\,.$$

Dette innebærer at

$$\frac{e^{\beta H_{N_i}(x_i)}}{\sum_{k=1}^{K} e^{\beta H_{N_i}(k)}}$$

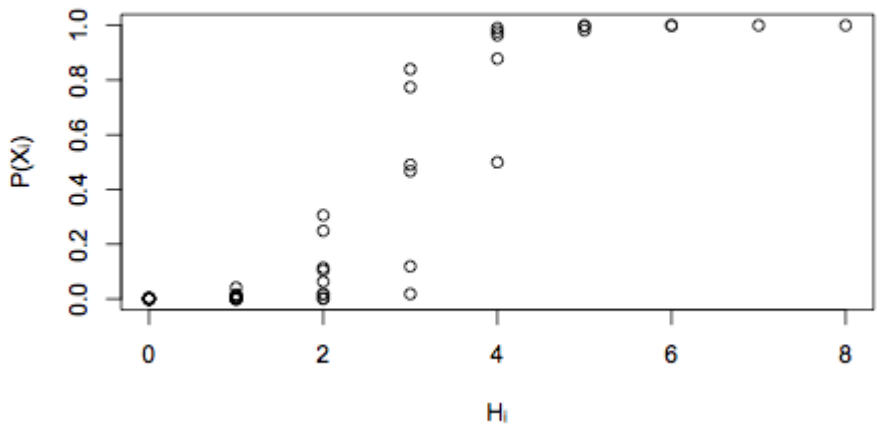
vil anta diskrete verdier, som varierer med β. For å gi et inntrykk av denne variasjonen har jeg i de følgende figurene plottet $P(X_i = x_i | x_j, j \in N_i)$ mot $H_{N_i}(x_i)$.



Figur 17: Figuren viser de distinkte verdiene P kan anta i hvert enkelt piksel i dersom β=0,7. Langs x har vi funksjonen $H_{N_i}(x_i)$, som gir antall av samme klasse som i i naboskapet N_i.

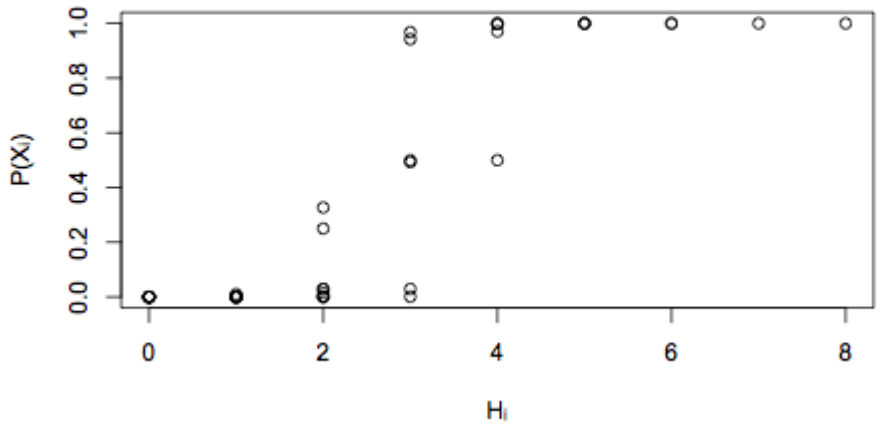
I denne figuren ser vi at vi har en samling av markørene for alle H. Funksjonen har en troverdig utvikling fra sannsynlighet lik 0 dersom man ikke har noen naboer av samme klasse. Ved H=4 har vi en sannsynlighet rundt 0,5. Og ved H=7 og 8 ligger den tett opp til 1,0.

På neste side er det en tilsvarende graf som viser sannsynlighetsverdiene for β = 2,0.



Figur 18: Figuren viser de distinkte verdiene P kan anta i hvert enkelt piksel i dersom β=2,0. Langs x har vi funksjonen H, som gir antall av samme klasse.For H=3 ser vi at vi får en bred spredning fra nesten 0,0 til 0,8.

Her ser vi at det er større spredning mellom markørene ved H = 2, 3 og 4. For de øvrige verdiene har vi tett samling. For H=0 og 1 er det tett samling rundt 0. Mens for H større enn 5 er det samling rundt 1,0. Det vi også ser er at H = 3 gir en stor variasjon rundt 0,5. Dette innebærer at dersom et piksel har 3 naboer, er den kontekstuelle sannsynligheten vel så avhengig av hvordan de resterende pikslene fordeler seg. Dette vil være av betydning når vi ser på antallet klasser.



Figur 19: Figuren viser de distinkte verdiene den kontekstuelle sannsynligheten kan ha. Dersom β = 3,5. Vist for ulike H.

Dersom vi øker β ytterligere ser vi at dette mønsteret blir enda tydeligere. Og vi ender opp med en

sannsynlighet som kan uttrykkes

$$P(X_i = x_i | x_j, j \in N_i) \approx \frac{\delta}{l} \ .$$

Hvor δ er lik 1 dersom klassen x_i har flest like naboer, og 0 ellers. Mens l er antallet klasser med like mange naboer som x_i .

5.3 Algoritmer for klassifisering

Fremdeles er målet med klassifiseringen å finne det bildet $x = \{x_1, x_2, ..., x_n\}$ som maksimerer aposteriori sannsynlighet (MAP). Nå må denne sannsynligheten uttrykkes både ved hjelp av fargeverdiene i bildet, og sannsynlighetsmodellen for MRF.

Dersom vi antar at de observerte fargeverdiene i hvert enkelt piksel er uavhengig av de andre fargeverdiene følger det at vår posteriori sannsynlighet $P(X = x|Y = y)$ også er en MRF.

Besag (1984) utnytter dette ved å sette opp sammenhengen:

$$P(X = x|Y = y) \propto P(Y = y|X = x)P(X = x)$$

Dubes et al. (1989) og Kato et al. (2006) forsøker å trekke denne sammenhengen videre inn i energifunksjonen fra kapittel 5.1

$$P(X = x|Y = y) \propto e^{-U(x)}$$

Her er normaliseringsleddet Z holdt utenom. Videre tenker de at $P(Y = y|X = x)$ utgjør klikkfunksjonen for enkeltstående klikker. Dersom vi beholder vårt naboskapsdefinisjon fra 5.2 ender vi opp med følgende uttrykk for energifunksjonen:

$$U(x) = \sum_{i=1}^{n} ln(\pi_{x_i} f_{x_i}(y_i)) + \sum_{i=1}^{n} H_{N_i}(x_i)$$

Å gå videre med denne likningen har ikke så mye for seg, man vil allikvel ikke komme frem til noe håndterbart resultat. I steden er det utviklet flere algoritmer som forsøker å finne den "beste" løsningen av en klassifisering basert på bruk av kontekstuell sannsynlighet.

Simulert herding

Simulert herding er en av disse. Etter Dubes et al. (1989) fungerer den ved at den finner en initiell $\hat{x_i}$, som er resultatet av å maksimere $f(y_i|\phi_{x_i})$. Deretter gjennomfører man herdingen i to løkker. En ytre løkke som sørger for å senke temperaturen i herdingen, og en indre løkke som gjennomfører gjentatte simuleringer.

For hver simulering dannes et bilde $\hat{z}$, ved å trekke en vilkårlig klasse for ett enkelt piksel mens resten beholdes fra $\hat{x}$. Deretter beregnes differansen

$$\Delta = P(X = \hat{z}|Y = y) - P(X = \hat{x}|Y = y) \ .$$

Dersom $\Delta > 0$ erstatter vi $\hat{x}$ med $\hat{z}$. Ellers tar vi hensyn til temperaturen T, ved at $\hat{x}$ erstattes med $\hat{z}$ gitt sannsynligheten $e^{\Delta/T}$. Den indre løkken gjennomløper bildet et gitt antall ganger.

Den ytre løkken sørger for å senke temperaturen T med funksjonen $g(t)$. I følge Dubes et al. (1989) brukte Geman & Geman (1984) funksjonen

$$T_{t+1} = \frac{ln(1+t)}{ln(2+t)} T_t \, .$$

En høy T sørger for at vi i starten prøver ut ulike varianter av bildet. Etter hvert som den ytre løkken sørger for at T går ned, vil betydningen av konteksten tilta. For å unngå å havne i lokale maksimum og heller oppsøke de globale er det viktig at funksjonen g(T) velges med omhu. Denne algoritmen har vist seg å gi gode resultater, men den kan være regneintensiv når størrelsen på bildet øker. Dette skyldes at den indre løkken gjennomløper hele bildet gjentatte ganger. I tillegg bør ikke den ytre løkka senke temperaturen for fort. Dubes et al. (1989) viser til at de hadde 300 iterasjoner av den ytre løkken og 200 gjennomløpinger av bildet i den indre løkken. Dette tilsier at denne algoritmen vil kreve mye regnekapasitet.

Maximising posterior marginals

En annen algoritme Maximising posterior marginals(MPM) har som hovedmål å minimere forventet antall feilklassifiseringer. Comer et al. (2000) viser til at dette er en bedre tilnærming enn MAP i forbindelse med klassifisering. Hovedforskjellen mellom disse to angrepsmåtene er at MAP ser på tiltroen til hele bildet under ett, mens MPM tar mer hensyn til sannsynligheten for riktige klassifiseringer i hvert enkelt piksel. Marroquin et al. (1987) viser at det å minimere forventet antall feilklassifiseringer er analogt med å maksimere

$$P(X_i = x_i|Y = y).$$

Dette uttrykket tilnærmes ved å gjennomføre gjentatte simuleringer av mulige resultater. Dermed har vi for hvert piksel en sannsynlighetsfordeling mellom de ulike klassene, og velger den klassen

med høyest sannsynlighet.
Algoritmen starter som i simulert herding med at vi setter klassetilhørigheten som resultatet av en klassifikasjon uten kontekst. Så gjør vi gjentatte simuleringer med bruk av kontekstuell sannsynlighet. Dette gjennomfører vi for hvert piksel i bildet. Først beregner vi $P(X_i = k|Y = y, X_j = x_j, j \in N_i)$ for hver $k \in K$. Med utgangspunkt i disse sannsynlighetene simulerer vi en ny klassetilhørighet x_i for pikselet. Så flytter vi oss til det neste pikselet og gjentar operasjonen. Når vi har gjennomført dette for hele bildet sitter vi igjen med et utfall $x^{(t)}$ som lagres. Samtidig tar vi med oss dette utfallet inn i en ny simulering av klassetilhørighet. Hele bildet simuleres T ganger. Til slutt tilnærmer vi

$$P(X_i = k|Y = y) \approx \frac{1}{T}\sum_{1}^{T} I(x_i^{(t)}, k)$$

hvor

$$I(a,b) = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases}$$

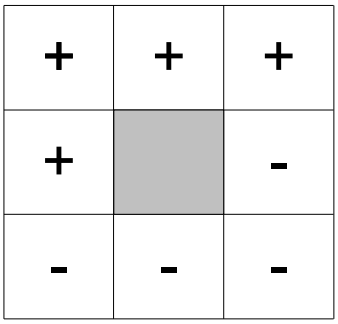
for alle $k \in K$. Dubes peker på at enkelte velger å vente med å lagre utfallene $x^{(t)}$ til man har oppnådd en stabil MRF.

Iterated conditional modes

Besag (1986) foreslo en annen algoritme, som tilnærmelse til MAP. Denne gav han betegnelsen Iterated Conditional Modes (ICM). Fordelen med den er at den er langt mindre regnekrevende, og i tillegg er den mindre tilbøyelig til å ende opp med å klassifisere alle pikslene som én enkelt klasse. Betegnelsen spiller på at vi benytter en betinget sannsynlighetsmodell for hvert enkelt piksel. Det vil si at vi ønsker å maksimere uttrykket

$$P(X_i = x_i|Y = y, X_j = x_j, j \in N_i)\,.$$

Som utgangsgpunkt for algoritmen bruker man gjerne en klassifisering basert på $\pi_{x_i} f(y_i, \phi_{x_i})$. Deretter går man inn i en løkke som fortløpende oppdaterer klassetilhørigheten for hvert piksel basert på en maksimering av uttrykket over. Nabopikslenes klassetilhørighet hentes utfra et skjema fremsatt av Green (1978). Han påviste at dette gir tilfredsstillende resultater.
Utgangspunktet er at vi starter i bildets øvre høyre hjørne, og gjennomløper det rad for rad. For hvert piksel benyttes klassetilhørigheten i nabopikslene tilsvarende figuren under. Hvert + representerer klassetilhørigheten i den inneværende iterasjonen, mens - betegner klassetilhørigheten i forrige iterasjon.



Figur 20: Skjema for hvilke tilstander som skal brukes ved oppdatering av klassetilhørighet i midtre piksel. + betegner tilstand i pågående iterasjon, - betegner tilstand i forrige iterasjon.

Antallet iterasjoner av denne løkken settes slik at resultatet konvergerer, eller at man ikke lenger får endringer i klassetilhørighet. Mine forsøk viser at dette gjerne skjer innen 10-20 iterasjoner.

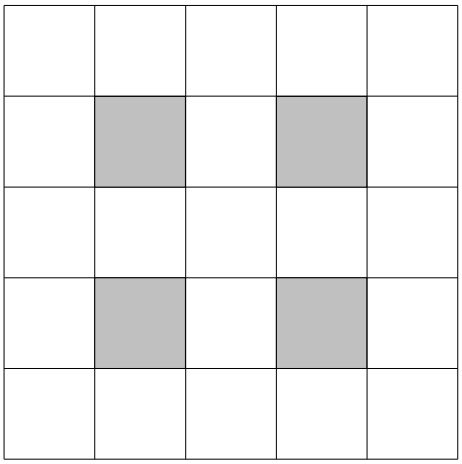
Den viktigste innsigelsen mot denne algoritmen er at den kan ha en tendens til å kjøre seg fast i lokale maksima fremfor å finne en global beste løsning.

5.4 Parameterestimering med pseudo-ML og EM

Ved innføringen av kontekst i sannsynlighetsmodellen har vi fått en ny parameter det er ønskelig å estimere. Dersom vi ønsker å lage en automatisert parameterestimering må vi ta for oss hele settet

$$\phi = \{\beta, \mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k, \pi_1, ..., \pi_k\} \ .$$

Dette er vanskelig å få til i én operasjon. For å estimere β må vi bruke x som en kjent størrelse. Det er den ikke den er avhengig av parametersettet $\{\mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k, \pi_1, ..., \pi_k\}$. Estimeringen av disse gjøres med en EM. I denne inngår den betingede sannsynligheten i hvert piksel. Dette vil nå være en kontekstuell sannsynlighet hvor β inngår. Dermed ser det ut til at dette må gjøres i en stegvis prosess.
For å estimere β fremsatte Besag (1974) et forslag om å bruke deler av et klassifiseringsresultat som inngangsverdi til en ML. Og ved gjentatte klassifiseringer var tanken å komme frem til et godt estimat for β gitt naboforholdene i bildet.
For å unngå avhengigheter mellom de x som brukes i estimeringen av β foreslo Besag (1974) et skjema for hvordan disse burde plukkes. Dette avhenger av hvilket naboforhold man bruker i den kontekstuelle modellen, men for et andreordens naboforhold kan man bruke mønster gitt av figuren på neste side.



Figur 21: Skjema som viser hvilke piksler (grått) som skal inngå i en ML-beregning.

For hver iterasjon teller vi opp antallet naboer. Dette brukes som inngangsverdier til en maksimering av sannsynlighetsfunksjonen vår. Resultatet av denne maksimeringen er vårt nye β-estimat.

Maksimal sannsynlighet finner vi ved å maksimere

$$L(\beta) = \prod_{i=1}^{N} \frac{e^{\beta H_{N_i}(x_i)}}{\sum_{k=1}^{K} e^{\beta H_{N_i}(k)}}.$$

Løsningen av en slik ligning finner vi gjerne ved å minimalisere den negative logaritmen.

$$-l(\beta) = -\sum_{i=1}^{N} \beta H_{N_i}(x_i) + log(\sum_{k=1}^{K} e^{\beta H_{N_i}(k)})$$

Når vi har funnet et estimat for β, skal vi estimere de øvrige parameterene. Dette gjør vi med en EM tilnærming, men nå inngår vår nye β i beregningen av sannsynlighet for klassetilhørighet i pikselet.

Algoritmen vi bruker kan beskrives med følgende oppsett.

1) Finn μ_k, Σ_k og π_k basert på treningssettet

2) Sett $\beta = 0$.

3) For hvert piksel beregn sannsynlighet, og plasser i klassen som gir størst sannsynlighet.

4) Tell naboskap og beregn ny β.

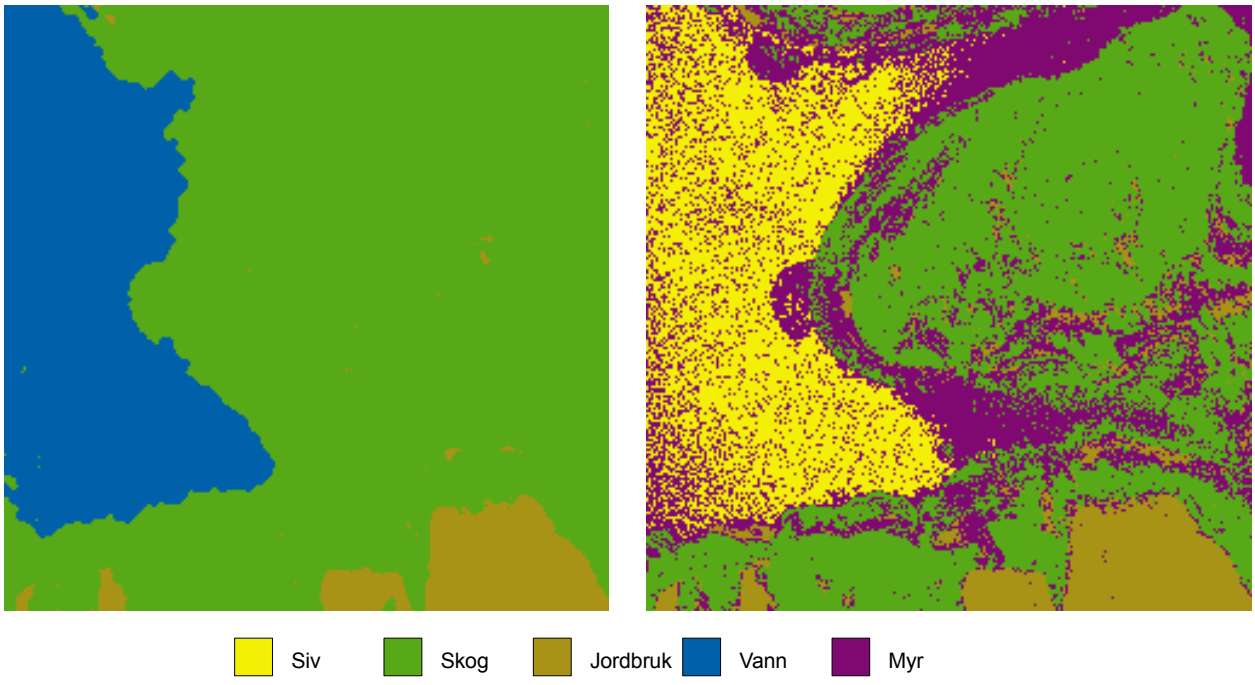
5) Beregn sannsynlighet for hver klassetilordning

6) Med sannsynligheten for klassetilordning finn nye μ_k og Σ_k.

7) Beregn ny π_k ved å telle opp antall i hver klasse

8) Gjenta 3) - 7) et gitt antall, eller inntil det ikke skjer endringer i klassetilordning

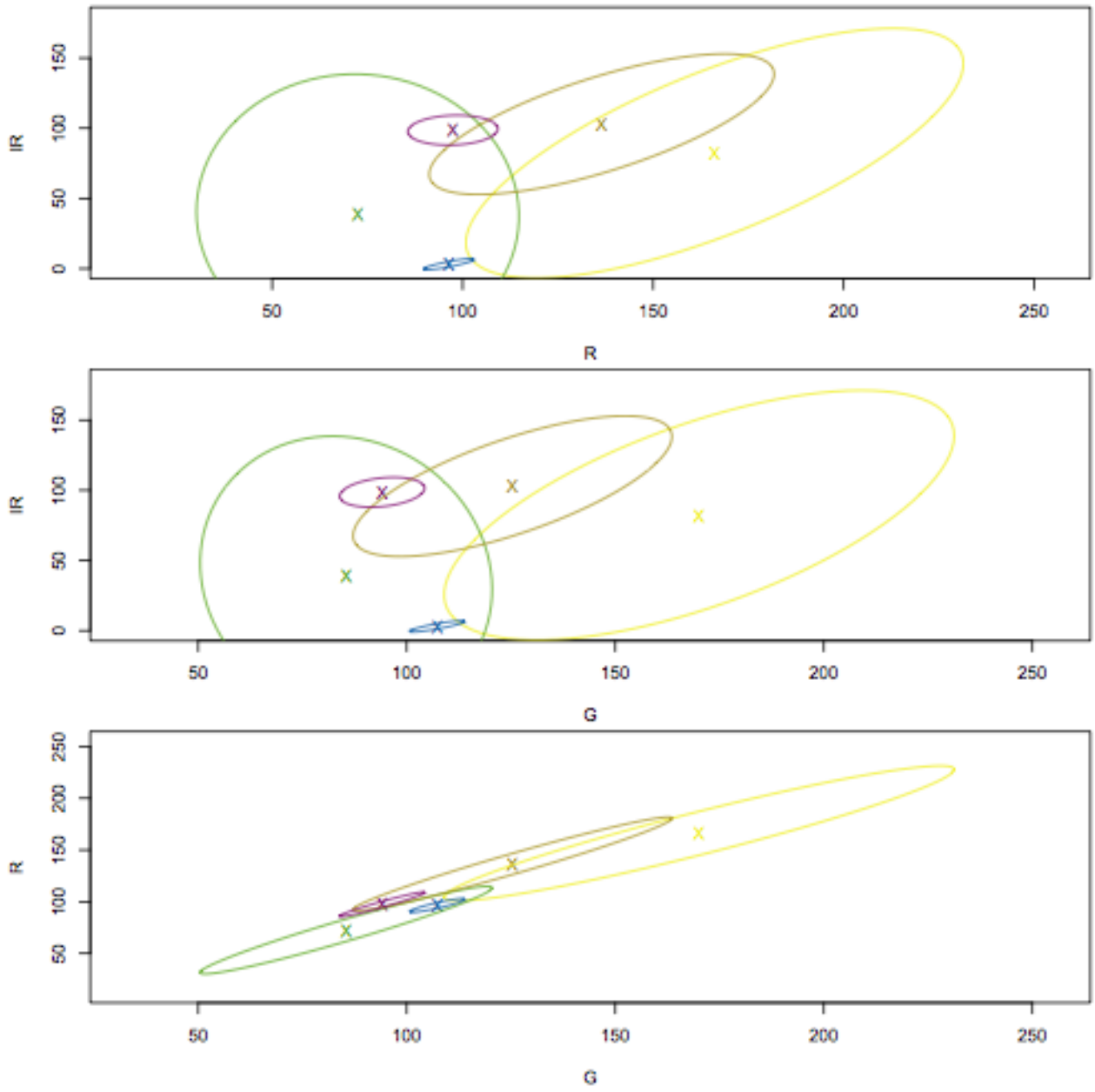
Denne fremgangsmåten får uønskede resultater for vårt datasett. Ved forsøk med 10 iterasjoner ender vi opp med at siv og myr forsvinner fra resultatsettet. Det endelige bildet er plassert til venstre i figuren under.



Figur 22: Bildet til venstre er resultat av klassifisering med bruk av kontekst, mens det til høyre viser resultat uten. Begge klassifiseringene har brukt EM for å tilpasse parameterene i de multivariable normalfordelingene.

For å sammenligne har jeg til høyre satt inn resultatet av å kjøre en automatisk klassifisering uten bruk av kontekst. Her ser vi at det er klare fellestrekk mellom de to bildene. Riktignok er det forskjellige klasser som forsvinner ut. Til venstre er det siv og myr, mens til høyre er det vann som forsvinner til fordel for siv.

I figuren under ser vi hvordan fordelingene til de nye klassene ser ut. Ellipsene er basert på parametersettene $\{\mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k\}$ etter 10 iterasjoner.



Figur 23: Fordeling av fargeverdier for de pikslene som inngår i klassene etter å ha gjentatt algoritmen 15 ganger. Vi ser at klassene myr og vann har blitt minimale

Dersom vi sammenligner med figur 5 er det tydelig at vann og myr har blitt betydelig mindre, mens de øvrige klassene har est i ulike retninger. Siden vann og myr har blitt betydelig mindre skulle vi forvente at også vann forsvinner fra resultatet, slik den gjør når vi ikke bruker kontekst. Dette kan skyldes at vi har større sammenhengende områder hvor pikslene har fargeverdier nær senteret for vann, og at de dermed trekkes mot vann av den kontekstuelle tilveksten. Mens for myr slår det motsatt ut, siden det er mindre sammenhengde områder med fargeverdier knyttet til senteret for myr. De kan vel så gjerne knyttes til senteret for skog eller jordbruk.

Dette var resultatet etter kjøring med 10 iterasjoner. Ved forsøk med ytterligere iterasjoner blir resultatet ytterligere forverret.

5.5 Bruk av pseudo-ML og fast α, μ og Σ

Dette resultatet viser at vi også ved bruk av kontekstuell klassifisering må holde et grep på parameterene $\{\mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k\}$. Derfor gjør vi noen forsøk med å bruke den α som gav best resultat i forrige kapittel. Algoritmen vår vil da unngå to av stegene i det forrige avsnittet.

1) Finn μ_k , Σ_k , Σ og π_k basert på treningssettet

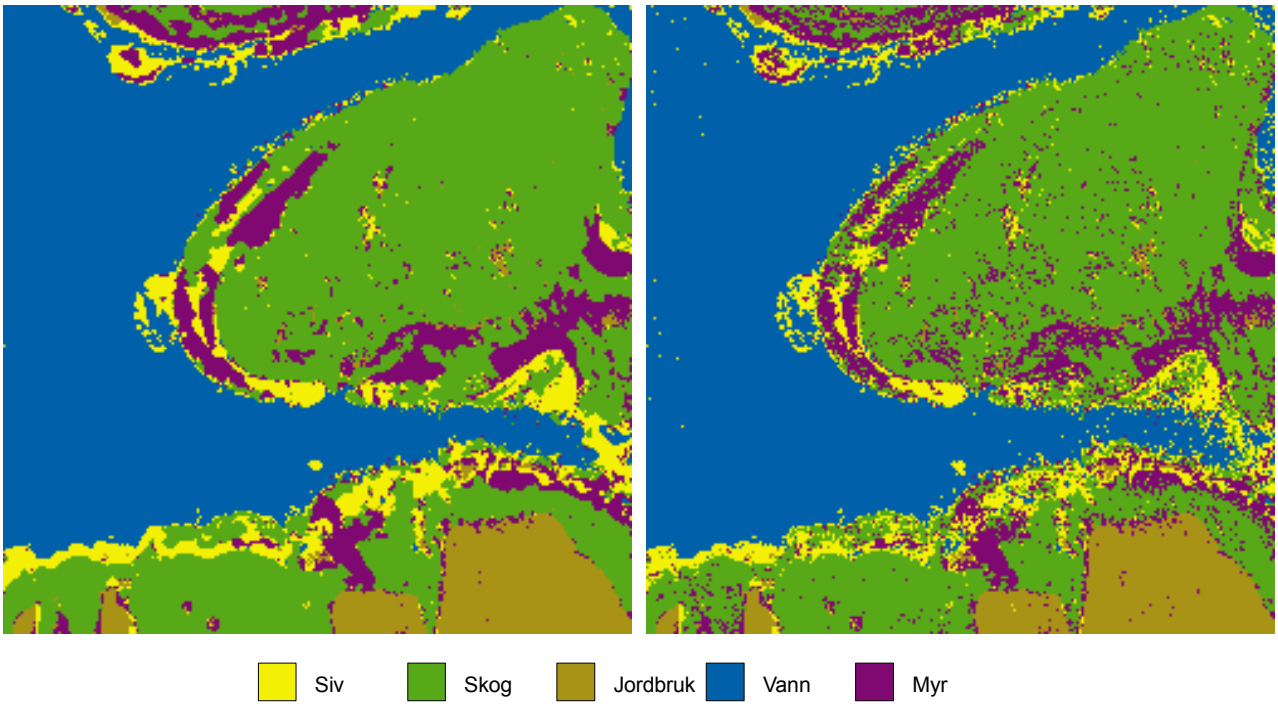
2) Sett $\beta = 0$.

3) For hvert piksel beregn sannsynlighet, og plasser i klassen som gir størst sannsynlighet.

4) Tell naboskap og beregn ny β .

5) Beregn ny π_k ved å telle opp antall i hver klasse

6) Gjenta 3) - 5) et gitt antall, eller inntil det ikke skjer endringer i klassetilordning



Figur 24: Bildet til venstre viser resultatet av å klassifisere med kontekst, mens det til høyre viser klassifisering uten. De øvrige parameterene har vært like i de to tilfellene. β ble satt ved en pseudo-ML metode og i bildet vises en β=0,78.

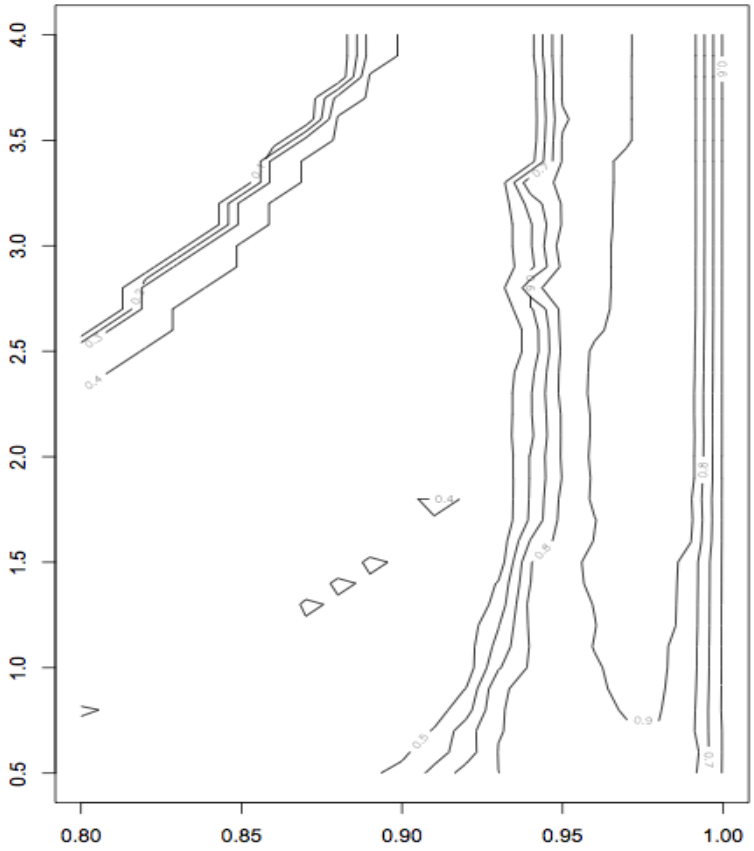
Her ser vi at vi er på vei mot det vi ønsker å oppnå med en klassifisering. Dersom vi sammenstiller dette med det opprinnelige bildet, ser vi at vi får avgrensede områder med en felles landskapstype. Fremdeles er det frittstående pikser vi kunne ønske oss å få fjernet. Og det er jo også områder som klassifiseres feil.
Vi ser at bruken av kontekst har bidratt til å fjerne enkeltpunkter, og gjort områdene mer helhetlige. Den β vi har kommet frem til er et resultat av det som gir størst sannsynlighet i bildet vårt sett under ett. Dersom vi ser på utviklingen av denne for hver iterasjon i algoritmen vår, ser vi at den konvergerer raskt mot 0,78.

Iterasjon	1	2	3	4	5	6	7	8	9	10
β	0,649	0,723	0,759	0,774	0,780	0,782	0,783	0,783	0,783	0,783

Tabell 3: Utvikling av estimert β ved 10 iterasjoner.

5.6 Bruk av feilmargin ved tilpasning av parametere

Her skal vi se på hvordan modellens sensitivitet og prediksjonsegenskaper for Siv utvikler seg når vi varierer α over intervallet 0,8 - 1 og β over intervallet 0,5 - 2,5. For hvert steg kjører vi en full klassifisering basert på μ_k og Σ_k hentet fra treningssettet, mens π_k endres for hver iterasjon.



Figur 25: Grafen viser klassifikatorens treffsikkerhet m.h.p. siv, for ulike verdier av α og β.

Maksimumsverdien for treffsikkerhet finner vi for α=0,98 og β=3,3.

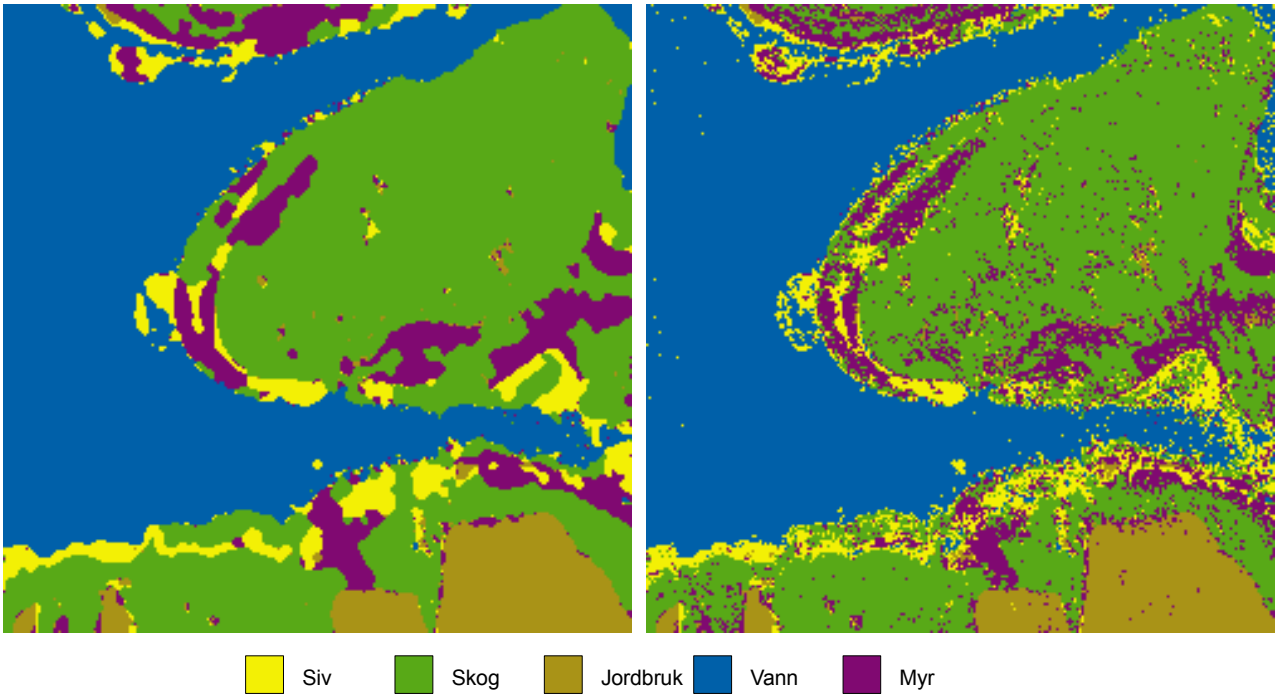
I tabellen under viser jeg verdiene for sensitivitet og presisjon for utvalgte verdier av α og β.

	Sensitivitet		
	α = 0,97	**α = 0,98**	**α = 0,99**
β = 2,0	0,963	0,968	0,984
β = 3,3	0,857	0,968	0,978
β = 4,0	0,778	0,963	0,978

Presisjon		
α = 0,97	**α = 0,98**	**α = 0,99**
0,963	0,948	0,886
0,993	0,989	0,930
0,993	0,989	0,930

Tabell 4: Sensitivitet og presisjon for siv i en modell med varierende α og β.

Av tabellen ser vi at sensitiviteten avtar med økende β, mens presisjonen øker. Samtidig ser vi at sensitiviteten øker med økende α, mens presisjonen avtar. Altså har de to variablene motsatt effekt på våre to koeffisienter.



Figur 26: Bildet til venstre viser klassifisering med bruk av kontekst, mens det til høyre viser tilsvarende metode uten bruk av kontekst. Bildet til venstre har α=0,98 og β=3,3.

Et annet moment som taler mot å øke β er at vi bruker ICM som metode for å finne maksimum. Denne kan oppsøke lokale maksimum fremfor å finne den globale. Det kan tenke at en annen algoritme vil være mer gunstig ved høye β.

Her ser vi et bilde som viser virkningen av å bruke kontekst i klassifiseringen. Begge bildene har α=0,98. Bildet til venstre har β=3,3.

Et annet poeng verdt å merke seg er at til tross for at Siv ser ut til å være overrepresentert i bildet til venstre, ved at det er ganske tykke striper i motsetning til i bildet til høyre, så har vi en $\pi_k = 0{,}06$ for siv i bildet til venstre. Mens den i det høyre bildet uten kontekst er 0,11.

6 Simulering

I fremstillingen så langt har vi basert oss på å validere resultatene mot treningssettet. Vi har ikke hatt et uavhengig kilde for å vurdere hvor gode metodene er i å kartlegge en vegetasjonstype. Det beste ville vært å ha feltmålinger, men vi kunne også løst det ved å dele opp treningssettet i to, ett for trening og ett for validering. Fremdeles vil det være basert på våre antakelser av hva som finnes i bildet.

En annen løsning er å lage vår egen fasit. Ikke i den betydningen av at det er et kart over de naturlige forholdene rundt Vansjø, men et selvkomponert kart over et område delt inn i ulike klasser. På bakgrunn av dette kartet kan vi simulere et flyfoto. I simuleringen legger vi inn antakelser om at fargeverdiene er hentet fra en multivariabel normalfordeling. Parameterene for de ulike klassene kan vi sette fritt. Jeg vil ta for meg to bilder. Ett hvor vi bruker parameterene fra forrige kapittel. Det andre bruker samme middelverdier, men kovariansmatrisene multipliseres med 2. Dette vil gi større overlapp mellom klassene, og større sjanse for feilklassifiseringer.

Det finnes rutiner for å simulere en multivariabel normalfordeling, men disse vil nødvendigvis generere kontinuerlige verdier. I bildesammenheng bruker vi diskrete verdier. Bildet vi har jobbet med så langt har hatt verdier i området $[0...255]$. Dette må vi ta hensyn dersom vi skal få en troverdig simulering. Dermed runder vi av til nærmeste heltall, og setter verdier under 0 lik 0, og verdier over 255 lik 255. Dette vil medføre at vi får en overrepresentasjon av 0-er i bildet vårt, noe som også synes å være tilfelle i det ordinære bildet. Dersom vi får en overrepresentasjon av 255 vil det være mer bekymringsfullt.

Med det simulerte flybildet blir oppgaven å teste ut ulike klassifikatorer, både uten og med kontekst, og se i hvilken grad de er i stand til å gjenskape fasiten. Vi vil også se på størrelsen av treningssettet, og hvor liten denne kan være før det får betydning for sluttresultatet.

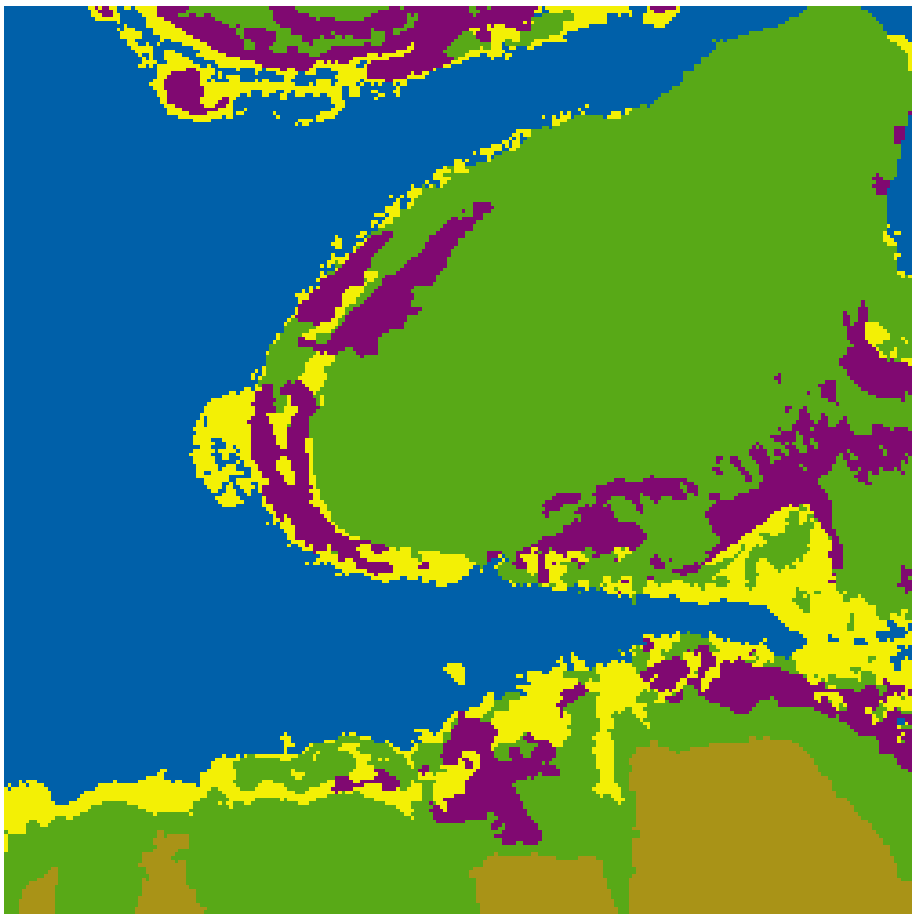
Som treningssett trekker vi et vilkårlig antall piksler i hele bildet og bruker vår kjennskap om klassetilhørighet. Deretter bruker vi metoden hvor parametere for de multivariable normalfordelingene estimeres på bakgrunn av treningssettet og holdes fast, mens π_k estimeres i en EM-algoritme. For den kontekstuelle klassifiseringen vil jeg bruke både faste β, og tilpasse den med pseudo-ML rutinen fra kapittel 5.3.

Siden vi trekker treningssettet vilkårlig har vi mulighet til å gjenta forsøket en rekke ganger, og se hvor stor variasjon vi får i sluttresultatet. Jeg valgte å gjennomføre alle forsøkene 100 ganger.

6.1 Simulert bilde

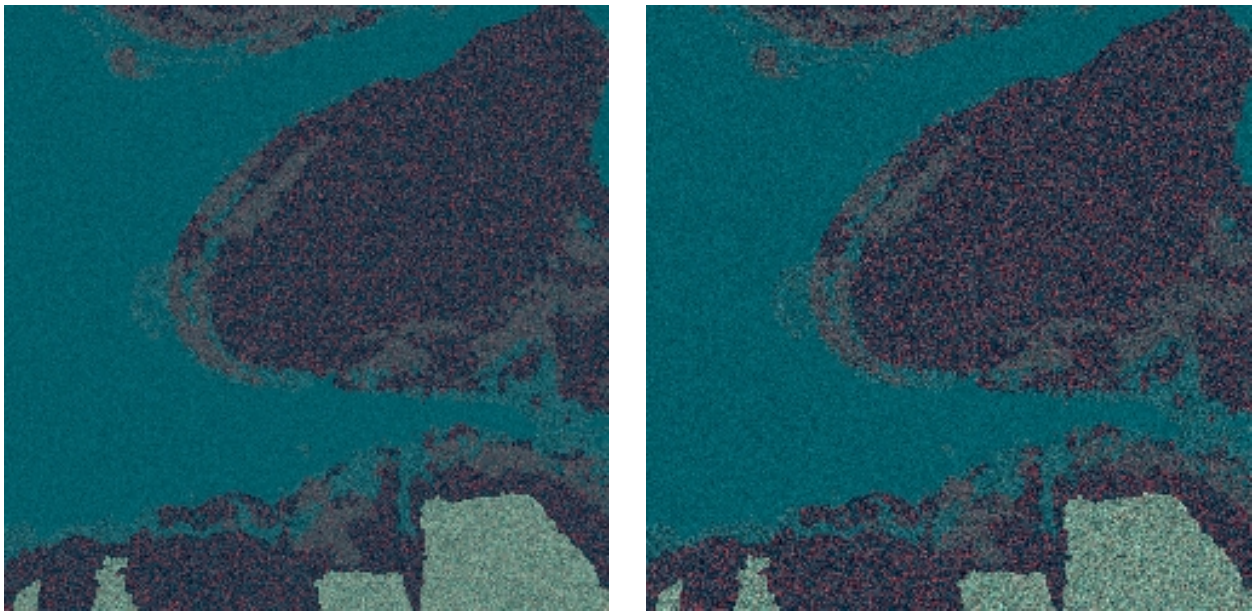
Fasiten jeg har utarbeidet er resultatet av en klassifisering av det opprinnelige bildet, med noen manuelle redigeringer. Dette har jeg valgt fordi det skal være gjenkjennelig, men utenom det er det viktig å holde det adskilt fra de øvrige resultatene i de foregående kapitlene.

Fasiten består av fem ulike klasser, og jeg bruker tilsvarende farger som i de foregående kapitlene.



Figur 27: Fasit for simuleringene i dette kapittelet. Bildet er laget på bakrgunn av et resultat fra forrige kapittel, men redigert på frihånd. Det består av fem klasser, som i forrige kapittel.

Bildene på neste side er generert ved at fargen i hvert piksel simuleres fra en multivariabel normalfordeling gitt av pikselets klassetilhørighet i bildet over.



Figur 28: Bilder som er simulert basert på fasiten og to multivariable normalfordelinger. Bildet t.v. er basert på parametere gitt av en klassifisering med β=2, mens det t. h. har en kovariansmatrise som er multiplisert med 2 i forhold til den t.v.

De ulike fordelingenes parametere er basert på resultatet i kapittel 5.6 med klassifisering gitt β=2,0. Parameterene ble estimert på bakgrunn av alle pikslene i bildet. Begge bildene benytter samme μ_k , men Σ_k er multiplisert med 2 i bildet til høyre.

6.2 Treningssett

I vårt opprinnelige bilde hadde vi markert 41% av bildet som treningssett. Dette er nok litt i overkant av hva som vil være tilfelle i praktisk bruk. Derfor vil jeg i dette kapittelet forsøke å se litt på størrelsen treningssettet bør ha for å få et godt estimat. Under vises en oversikt over andelen av de respektive klassene i fasiten. Dette bør samsvare med de π_k vi får etter hver klassifisering.

Klasse	Andel fasit
Gul	0,08
Grønn	0,39
Oransje	0,06
Blå	0,36
Lilla	0,09

Tabell 5: Fordeling mellom klassene i fasiten.

Jeg ønsker å se hvordan resultatet blir dersom jeg bruker et treningssett tilsvarende 20 % av pikslene i det opprinnelige bildet kontra å bruke 5 %, eller 0,5 % Dette kan vi få frem ved å gjenta forsøket med ulike treningssett et gitt antall ganger, og se på variasjonen i hvilken klasse pikslene havner i etter klassifiseringen.

For hver kjøring får vi et klassifiseringsresultat eller utfall $x^j = \{x_1^j, ..., x_n^j\}$. Av våre M utfall finner vi frem til sannsynligheten for hver klasse i pikselet med

$$V_{i,k} = 100\frac{m_{i,k}}{M}$$

hvor $m_{i,k}$ er antall utfall med klassen k i pikselet i. Dette kan settes sammen til en vektor for hele bildet med

$$V = \{V_{1,j_1}, V_{2,j_2}, ..., V_{n,j_n}\} .$$

Det er to nærliggende valg for $j = \{j_1, j_2, ..., j_n\}$. Vi har V_M som svarer til å velge

$$j = z \text{ hvor } z = \{z_i = argmax_{k=1}^{K} V_{i,k}\}$$

Den andre V_F gir oss sannsynligheten for å treffe fasiten i hvert piksel. Her er

$$j = f \text{ hvor } f \text{ er fasiten.}$$

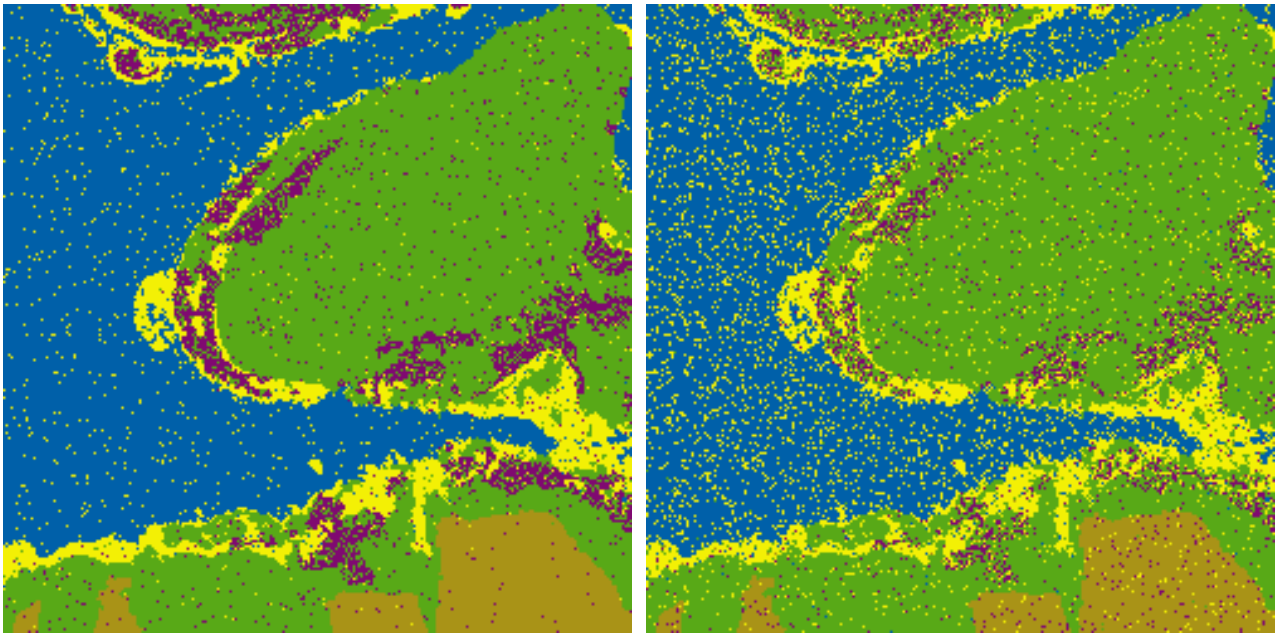
I mine forsøk har jeg brukt M = 100. Sluttresultatet som presenteres for hver kjøring refererer til bildet z. Altså den klassen som forekommer oftest i sluttresultatet for hvert piksel. Dette resultatet vil antakeligvis ikke være så avhengig av størrelsen på treningssettet, men det er å forvente at variasjonen i hvert piksel er større når størrelsen på treningssettet er mindre.

6.3 Klassifisering uten kontekst

Først skal vi se på klassifisering basert på at parameterene til de multivariable normalfordelingene hentes fra et treningssett, mens π_k tilpasses ved å iterere 10 ganger. Klassetilhørigheten i treningssettet er kjent. Dette ble gjennomført for et treningssett på 0,5 %, 5% og 20%.

Samme forsøk ble gjennomført på de to simulerte bildene. I det ene er Σ_k tilsvarende som i resultatet etter en klassifisering med $\beta = 2,0$, mens i det andre har vi multiplisert Σ_k med to.

På neste side viser vi bildet z. Dette svarer til den klassen som forekommer oftest i hvert piksel. Som vi antok i innledningen viste det seg at denne var uavhengig av størrelsen på klassen, derfor viser jeg kun de bildene som fremkommer når treningssettet er minst, altså 0,5 % av alle pikslene.



Figur 29: Resultatet å kjøre en ikke-kontekstuell klassifikator på to simulerte bilder. Til venstre Σ og til høyre 2Σ. Bildene viser den klassen som forekommer oftest for hvert piksel i et forsøk med 100 gjennomganger. For hver runde trekkes et vilkårlig treningssett på 0,5 % av pikslene.*

Her ser vi at begge bildene har endel støy. Det til høyre har vesentlg mer enn det til venstre, og dette er helt i overensstemmelse med våre antakelser. Siden bildet til høyre har vesentlig mer overlapp mellom fordelingene enn det til venstre.

I forvirringsmatrisene på neste side står Klassifisert for bildet z, mens Treningssett representerer fasiten. Klassene betegnes med navnet på fargen fremfor de navnene jeg brukte tidligere. Også i forbindelse med forvirringsmatrisen er forskjellene mellom de ulike treningssettene så små at jeg kun viser de to som fremkommer av et treningssett basert på 0,5 % av pikslene.

		Treningssett						
		Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Klassifisert	Gul	5044	167	8	729	161	6109	82.57%
	Grønn	95	24108	1	1	1578	25783	93.50%
	Oransje	6	0	3910	0	43	3959	98.76%
	Blå	121	3	0	21897	0	22021	99.44%
	Lilla	96	625	43	0	3864	4628	83.49%
	Sum	5362	24903	3962	22627	5646	62500	
	Sens.	94.07%	96.81%	98.69%	96.77%	68.44%		
						Treffpros.	94.12%	

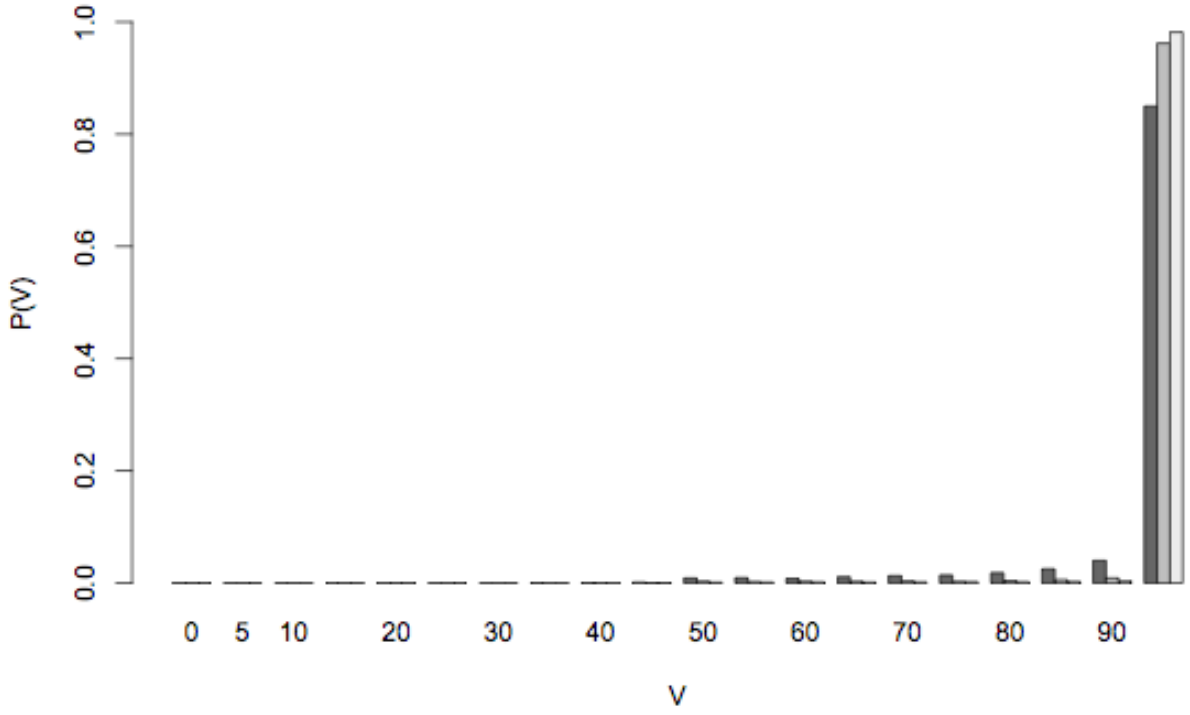
Figur 30: Forvirringsmatrise for hele bildet. Klassene er navngitt etter fargen. Dette er resultatet av å plukke ut den klassen som er hyppigst forekommende i en klassifisering av et bilde basert på en Σ.

		Treningssett						
		Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Klassifisert	Gul	4649	835	101	2716	710	9011	51.59%
	Grønn	336	23603	6	42	2834	26821	88.00%
	Oransje	41	0	3686	0	181	3908	94.32%
	Blå	206	41	0	19869	1	20117	98.77%
	Lilla	130	424	169	0	1920	2643	72.64%
	Sum	5362	24903	3962	22627	5646	62500	
	Sens.	86.70%	94.78%	93.03%	87.81%	34.01%		
						Treffpros.	85.96%	

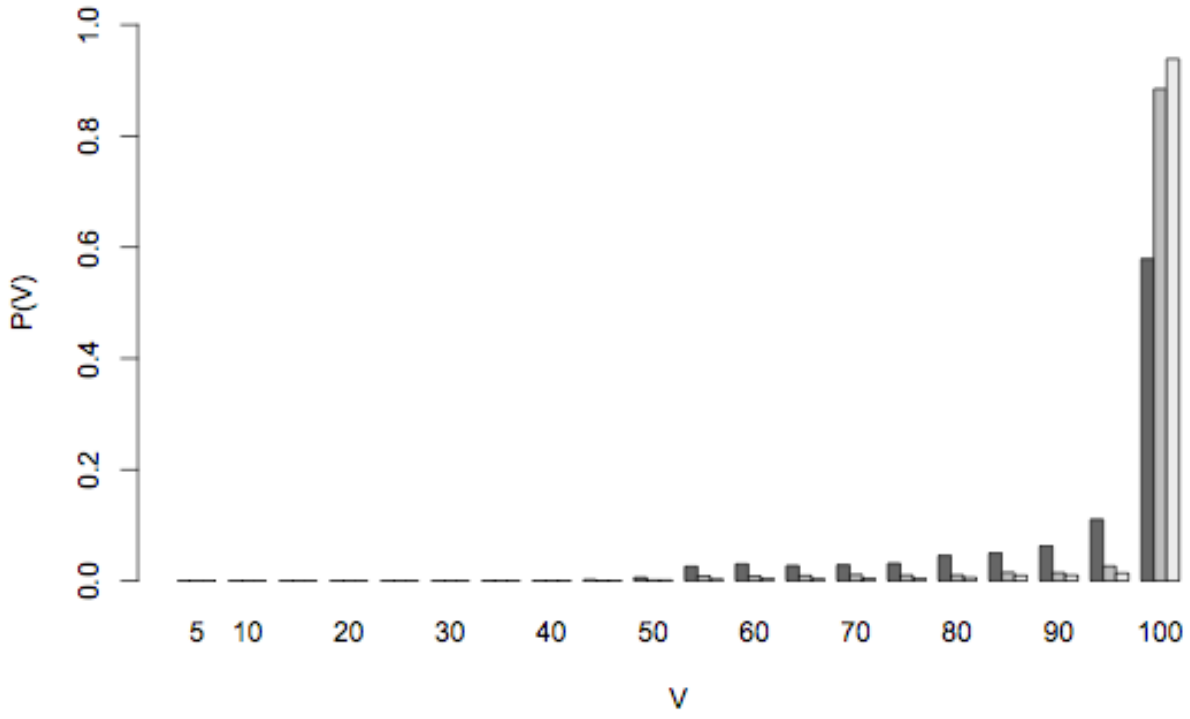
*Figur 31: Forvirringsmatrise for hele bildet. Klassene er navngitt etter fargen. Dette er resultatet basert på den hyppigst forekommende klassen i 100 repetisjoner av en klassifisering av et simulert bilde med 2*Σ*

Av forvirringsmatrisen ser vi at vi får vesentlig bedre resultat når kovariansen er Σ_k fremfor $2\Sigma_k$. Som forventet ser vi en gjennomgående forverring av resultatet når vi øker kovariansen i det simulerte bildet. Dette gjelder både sensitiviteten og presisjonen.

Under og på de neste sidene vil jeg presentere histogrammer som viser fordelingen til V_M. $V_{M,\{i\}}$ er sannsynligheten uttrykt i prosent for utfallet $x_i = z_i$ blant våre forsøk. Siden vi har 5 klasser vil denne ha en nedre grense på 20.



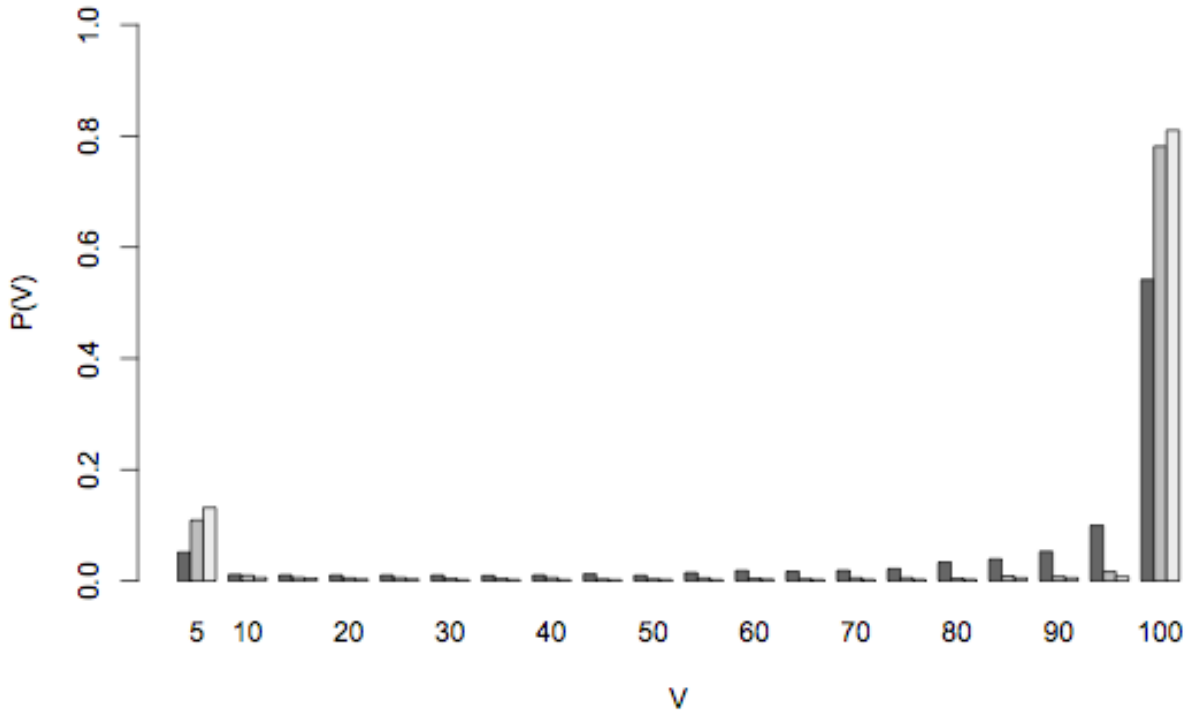
Figur 32: Dette plottet er en sammenstilling av histogram, som viser variasjon i utfall med enkel Σ og ingen kontekst. Søylene svarer til ulike størrelser på treningssettet, f.v. 0,5% - 5% - 20%. Søylene viser andelen av piksler P(V) innenfor hvert intervall av V. V svarer til prosentandelen av utfallene hvor man ender opp med samme klasse i pikselet. Dette plottet viser resultatet for klassen med størst V. Siden vi har 5 klasser må denne være større enn 20.



*Figur 33: Dette plottet er en sammenstilling av histogram, som viser variasjon i utfall med enkel 2*Σ og ingen kontekst. Søylene svarer til ulike størrelser på treningssettet, f.v. 0,5% - 5% - 20%. Søylene viser andelen av piksler P(V) innenfor hvert intervall av V. V svarer til prosentandelen av utfallene hvor man ender opp med samme klasse i pikselet. Dette plottet viser resultatet for klassen med størst V. Siden vi har 5 klasser må denne være større enn 20.*

Ved å sammenlikne de to histogrammene ser vi at vi har svært liten variasjon i utfallet av klassifiseringer. Når treningssettet er på 5% eller 20% av hele bildet ender vi opp med det samme klassifiseringsresultatet. Når treningssettet kun utgjør 0,5 % av bildet ser vi litt variasjon i klassifiseringsresultatet, og denne er mest markant når spredningen i bildet er størst.

Vi kan også se på hvordan fordelingen er for å ende opp med rett klassifisering. Altså at utfallet vårt samsvarer med fasiten. Denne har vi gitt betegnelsen V_F. På neste side har jeg satt inn et tilsvarende plott som tidligere for fordelingen til V_F. Tilsvarende som i forrige plott er mønsteret likt for Σ_k og $2 * \Sigma_k$, men utslagene større når kovariansmatrisen økes. Derfor viser jeg bare plottet som svarer til $2 * \Sigma_k$.

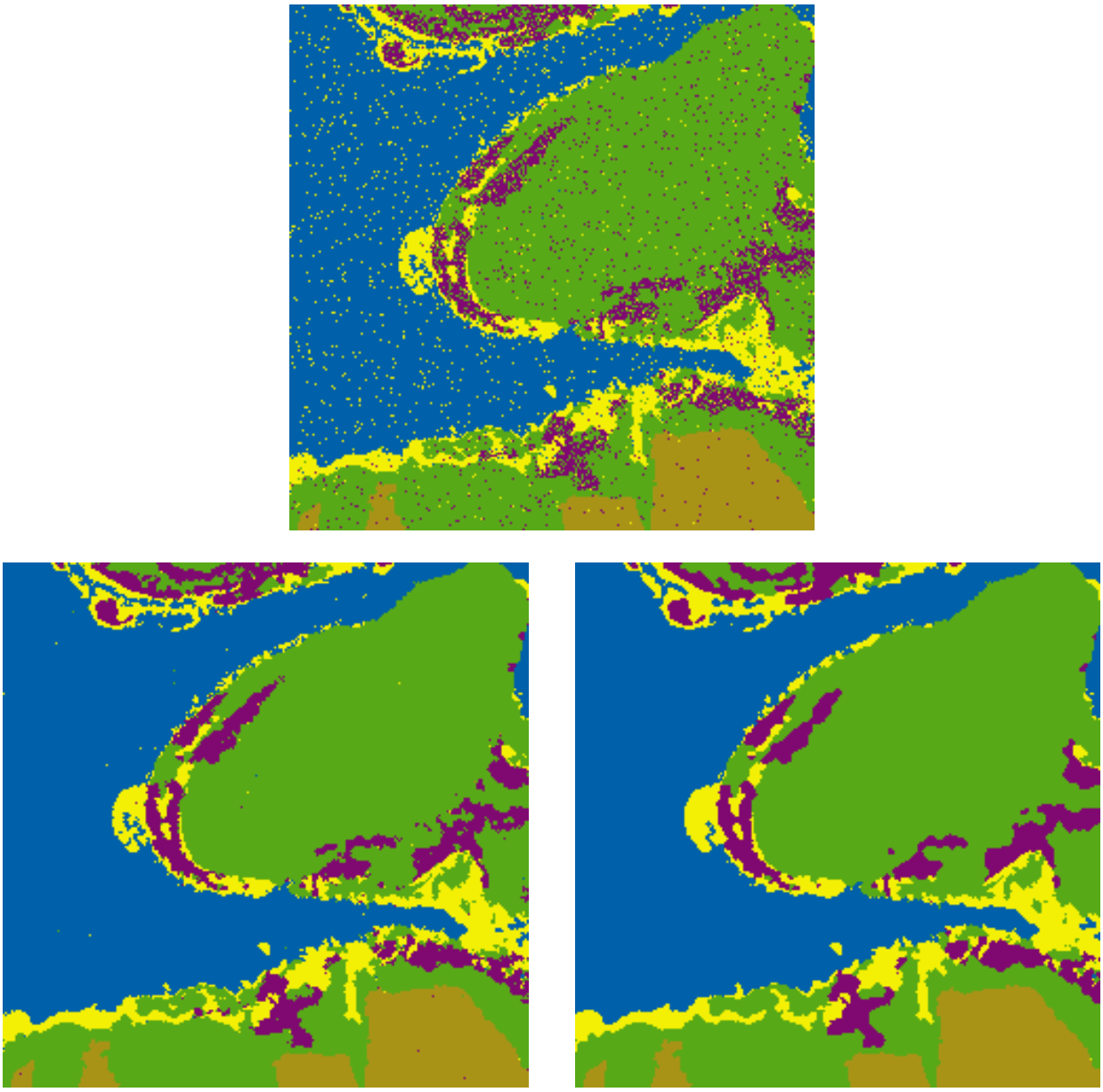


Figur 34: Plott som viser fordelingen P(V) av V_F. V_F svarer til sannsynligheten for å treffe fasiten i hvert piksel. Søylene svarer til ulike størrelser på treningssettet, f.v. 0,5% - 5% - 20%. Vi ser tilsvarende mønster som i forrige plott at det er liten variasjon i utfallene, og at den øker med avtakende treningssett. Det vi også ser er at vi har en topp for V = 0-5. Dette svarer til at enkelte av pikslene i fasiten aldri eller sjelden blir klassifisert riktig. Denne andelen avtar dersom vi minsker treningssettet.

Vi ser samme mønster som for V_M. En stor andel av pikslene havner i intervallet 95 - 100. Og denne andelen øker med økende treninssett. Det som er nytt i denne figuren er at vi har en topp også i intervallet 0-5. Dette svarer til at enkelte av pikslene blir klassifisert feil i alle eller nesten alle gjennomgangene. Denne andelen avtar dersom vi bruker mindre treningssett.

6.4 Kontekstuell klassifisering

Nå skal vi se om vi får en forbedring dersom vi bruker kontekstuell sannsynlighet i klassifiseringen. Først vil jeg presentere resultatene vi oppnådde ved å klassifisere et bilde med kovarianser lik de vi oppnådde som klassifiseringsresultat i forrige kapittel. Forsøkene er gjennomført med β=0,6 og β=2,0. Som i forrige avsnitt har jeg for hver klassifikator gjennomført 100 forsøk med et vilkårlig treningssett. Størrelsen på treningssettet har vært på 0,5 %, 5 % og 20 % av det opprinnelige bildet. På neste side presenterer jeg bilder som viser resultatet ved ikke å bruke kontekst, ved å bruke kontekst og β=0,6 og ved å bruke β=2,0. For alle bildene som vises her har størrelsen på treningssettet vært 5% av bildets piksler.



Figur 35: Bildene viser den klassen som forekommer oftest ihvert piksel. Øverst vises klassifisering uten kontekst. Nede t.v. vises klassifisering med β=0,6 og t.h. β=2,0. For alle bildene har størrelsen på treningssettet vært 5% av alle pikslene.

Som vi ser er det langt mindre støy når vi bruker kontekst i klassifiseringen. Det er også mulig å se at bildet til høyre er mer utglattet enn det til venstre. På neste side har jeg satt inn forvirringsmatrisene som viser forskjellene mellom disse bildene og fasiten.

Klassifisert	Treningssett						
	Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Gul	5062	185	8	793	169	6217	81.42%
Grønn	85	24023	1	1	1493	25603	93.83%
Oransje	6	0	3910	0	42	3958	98.79%
Blå	116	3	0	21833	0	21952	99.46%
Lilla	93	692	43	0	3942	4770	82.64%
Sum	5362	24903	3962	22627	5646	62500	
Sens.	94.41%	96.47%	98.69%	96.49%	69.82%		
					Treffpros.	94.03%	

Klassifisert	Treningssett						
	Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Gul	5242	6	2	47	37	5334	98.28%
Grønn	44	24877	1	5	782	25709	96.76%
Oransje	2	0	3953	0	7	3962	99.77%
Blå	42	2	0	22575	0	22619	99.81%
Lilla	32	18	6	0	4820	4876	98.85%
Sum	5362	24903	3962	22627	5646	62500	
Sens.	97.76%	99.90%	99.77%	99.77%	85.37%		
					Treffpros.	98.35%	

Klassifisert	Treningssett						
	Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Gul	5172	11	1	265	25	5474	94.48%
Grønn	127	24845	1	2	1053	26028	95.45%
Oransje	0	0	3960	0	0	3960	100.00%
Blå	20	0	0	22360	0	22380	99.91%
Lilla	43	47	0	0	4568	4658	98.07%
Sum	5362	24903	3962	22627	5646	62500	
Sens.	96.46%	99.77%	99.95%	98.82%	80.91%		
					Treffpros.	97.45%	

Figur 36: Forvirringsmatriser som viser forbedringen i resultat med bruk av kontekst. Øverst er resultatet av en ikke-kontekstuell, i midten vises en med β=0,6 og nederst med β=2,0.

Det vi ser av disse er at vi får en forbedring av resultatet ved å bruke β=0,6, men deretter en forverring med β=2,0. Dette vises tydelig på den lilla klassen, hvor en stor andel av pikslene blir

klassifisert som grønn.

Resultatene jeg har vist så langt har vært basert på et treningssett med 5% av den totale pikselmengden. I kapittel 6.3 viste det seg at sluttresultatet var like godt for en klassifisering basert på et treningssett bestående av 0,5% av pikslene, men at variasjonen i sluttresultat økte.
Når vi introduserer kontekstuell sannsynlighet viser det seg at denne sammenhengen fortoner seg litt annerledes. Her gir et treningssett på 0,5% av pikslene dårligere resultat enn et med 5%, mellom 5% og 20% er forholdet det samme.

Dette kan vi se av følgende forvirringsmatriser, som viser resultatet av klassifiseringer med β=0,6.

		Treningssett					
		Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Klassifisert	Gul	5235	4	2	40	39	5320	98.40%
	Grønn	55	24886	1	7	1117	26066	95.47%
	Oransje	2	0	3952	0	7	3961	99.77%
	Blå	44	2	0	22580	0	22626	99.80%
	Lilla	26	11	7	0	4483	4527	99.03%
	Sum	5362	24903	3962	22627	5646	62500	
	Sens.	97.63%	99.93%	99.75%	99.79%	79.40%		
						Treffpros.	97.82%	

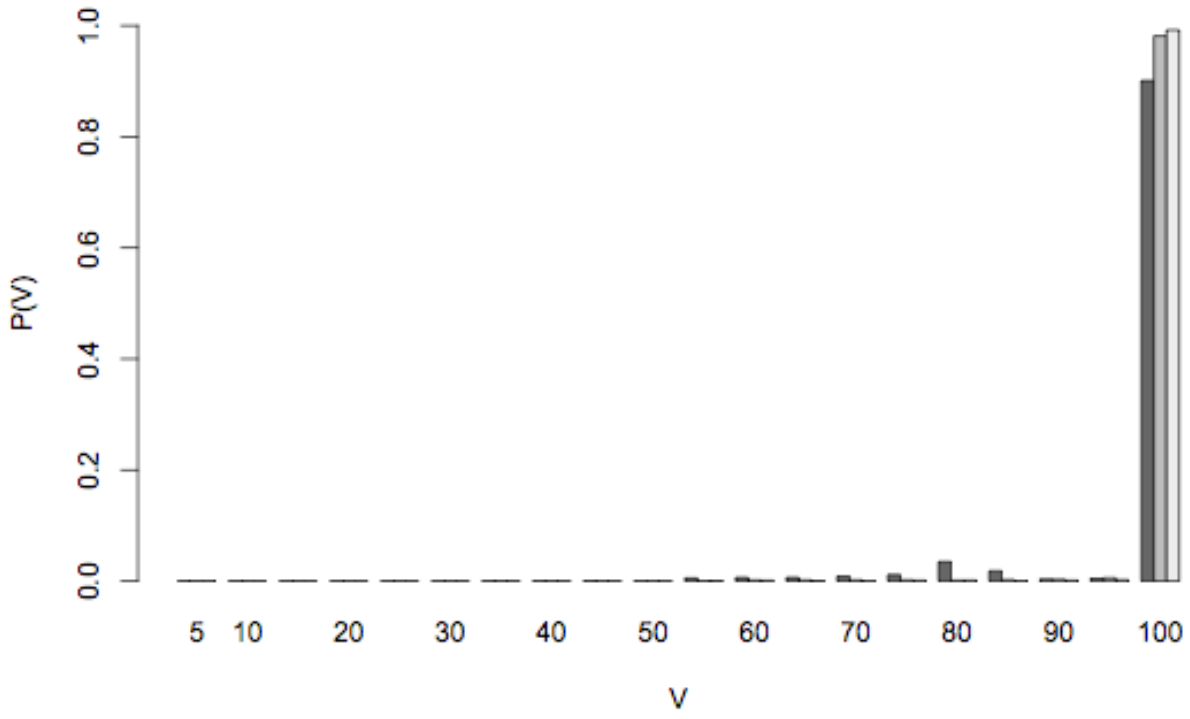
Figur 37: Forvirringsmatrise som viser Klassifisert (z) mot Treningssett (f) når β=0,6 og treningssettet tilsvarer 5% av totalbildet.

		Treningssett					
		Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Klassifisert	Gul	5242	6	2	47	37	5334	98.28%
	Grønn	44	24877	1	5	782	25709	96.76%
	Oransje	2	0	3953	0	7	3962	99.77%
	Blå	42	2	0	22575	0	22619	99.81%
	Lilla	32	18	6	0	4820	4876	98.85%
	Sum	5362	24903	3962	22627	5646	62500	
	Sens.	97.76%	99.90%	99.77%	99.77%	85.37%		
						Treffpros.	98.35%	

Figur 38: Forvirringsmatrise som viser Klassifisert (z) mot Treningssett (f) når β=0,6 og treningssettet tilsvarer 20% av totalbildet.

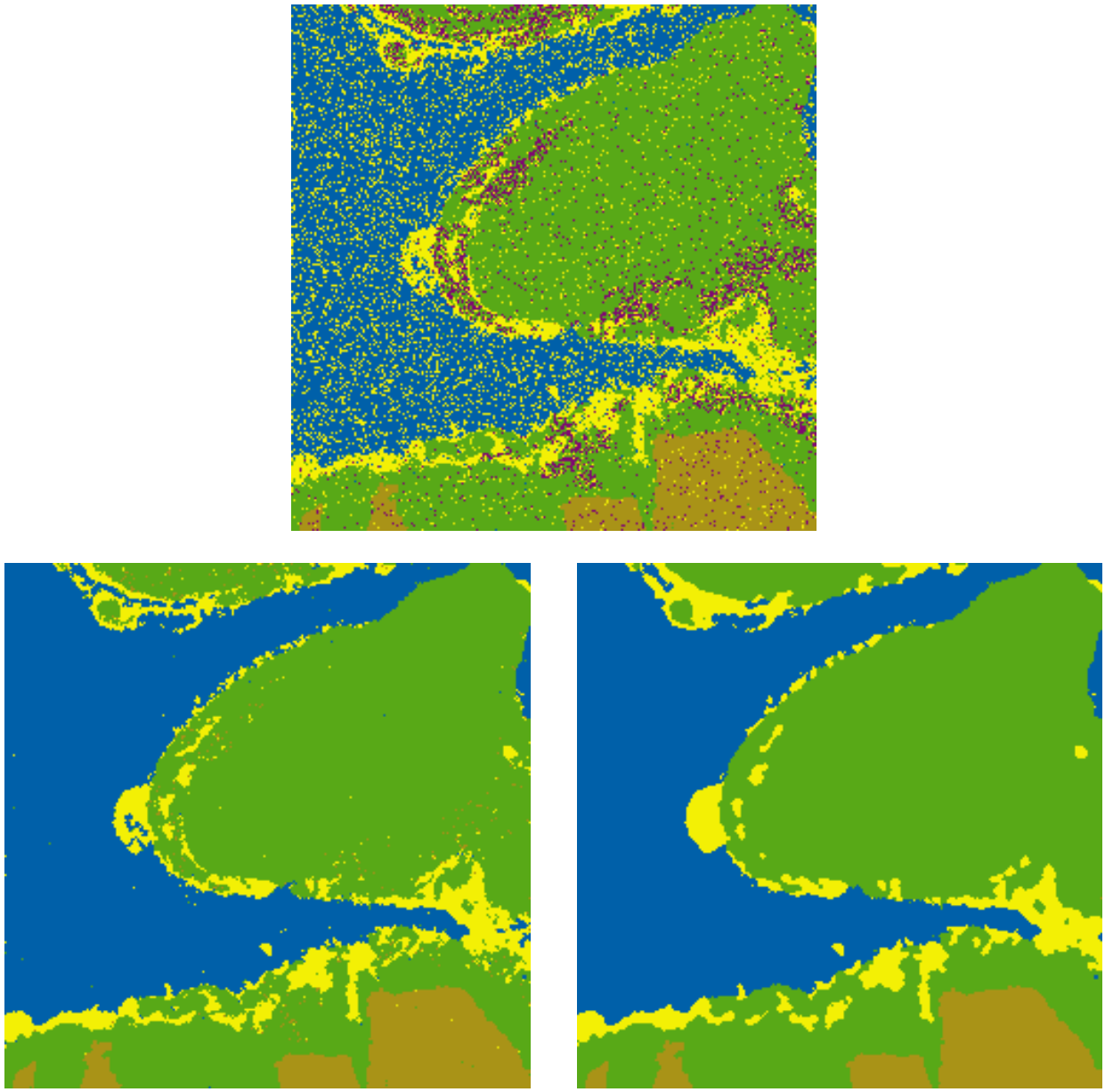
Her ser vi at vi får en minimal fordedring av resultatet ved å øke treningssettet fra 5% til 20%.

Histogrammet under viser variasjonen i hvilken klasse hvert enkelt piksel tilordnes er for β=0,6 lik den for ikke-kontekstuell klassifisering. D.v.s. en ørliten større variasjon når treningssettet er mindre. For β=2,0 inntreffer et annet forhold. Da så vi av forvirringsmatrisen at vi fikk en forverring i resultatene. I histogrammene medfører denne forverringen at vi har fått en liten topp på histogrammet ved V=80.



Figur 39: Histogram som viser fordelingen for V_M når β=2,0. Andelen for å havne i intervallet 95-100 er større enn for ikke-kontekstuell. Samtidig har vi fått en ny topp i intervallet 80-85.

Når vi tar for oss bildet hvor vi har brukt kovariansmatriser som er $2 * \Sigma_k$ inntrer et annet forhold. Nå blir klassen lilla så overlappende med de andre klassene, hovedsakelig den grønne, at den ikke er representert i noen av utfallene. Dette er gjennomgående for alle testene uavhengig av valg av β, eller størrelse på treningssettet.



Figur 40: Bildene viser de hyppigst forekommende klassene etter 100 klassifiseringer. Utgangspunktet er et bilde med kovariansmatrise $2 * \Sigma_k$. Bildene svarer til øverst en ikke-kontekstuell klassifikator, nede t.v. β=0,6 og t.h. β=2,0.Vi ser at den lilla klassen forsvinner når vi bruker den kontekstuelle tilnærmingen.

Her ser vi at klassen lilla har blitt helt radert ut. Utenom dette virker resultatet å samsvare med bildene i figur 35. Ved å bruke β=2,0 får vi færre frittstående piksler og mer utglattede grenser mellom klassene.

Klassifisert	Treningssett: Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Gul	4756	989	103	3802	741	10391	45.77%
Grønn	274	23341	6	23	2607	26251	88.91%
Oransje	34	0	3679	0	169	3882	94.77%
Blå	175	31	0	18802	1	19009	98.91%
Lilla	123	542	174	0	2128	2967	71.72%
Sum	5362	24903	3962	22627	5646	62500	
Sens.	88.70%	93.73%	92.86%	83.10%	37.69%		
					Treffpros.	84.33%	

Klassifisert	Treningssett: Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Gul	4954	21	5	105	240	5325	93.03%
Grønn	323	24876	10	17	5169	30395	81.84%
Oransje	21	0	3947	0	237	4205	93.86%
Blå	64	6	0	22505	0	22575	99.69%
Lilla	0	0	0	0	0	0	NaN%
Sum	5362	24903	3962	22627	5646	62500	
Sens.	92.39%	99.89%	99.62%	99.46%	0.00%		
					Treffpros.	90.05%	

Klassifisert	Treningssett: Gul	Grønn	Oransje	Blå	Lilla	Sum	Pres.
Gul	4587	18	0	393	79	5077	90.35%
Grønn	746	24885	5	23	5565	31224	79.70%
Oransje	0	0	3957	0	2	3959	99.95%
Blå	29	0	0	22211	0	22240	99.87%
Lilla	0	0	0	0	0	0	NaN%
Sum	5362	24903	3962	22627	5646	62500	
Sens.	85.55%	99.93%	99.87%	98.16%	0.00%		
					Treffpros.	89.02%	

Figur 41: Forvirringsmatriser for klassifiseringer med et treningssett tilsvarende 5 % av bildets piksler. Øverst vises resultatet av ikke å bruke kontekst, i midten bruk av kontekst med β=0,6 og nederst β=2,0.

Av de to øverste forvirringsmatrisene ser vi en klar forbedring av resultatet for de fleste klassene.

Det klare unntaket er klassen lilla, som er radert ut. De øvrige klassene får en bedring i sensitivitet, mens klassene grønn og oransje får dårligere presisjon. Dette skyldes i all hovedsak at de trekker til seg piksler som skulle vært klassifisert i den lilla klassen. Den totale treffsikkerheten forbedres med bruk av kontekst, men når vi øker β til 2,0 får vi en liten nedgang.
Variasjonen i utfallene viser samme mønster som når kovariansmatrisen settes lik Σ_k.

6.5 Tilpasning av β med pseudo-ML

For begge bildene gjorde jeg også forsøk med å tilpasse β ved hjelp av algoritmen i kapittel 5.5. Den innebærer at vi trekker et vilkårlig treningssett og estimerer parameterene μ_k og Σ_k. Deretter tilpasser vi π_k og β iterativt. Dette forsøket ble gjentatt 100 ganger.

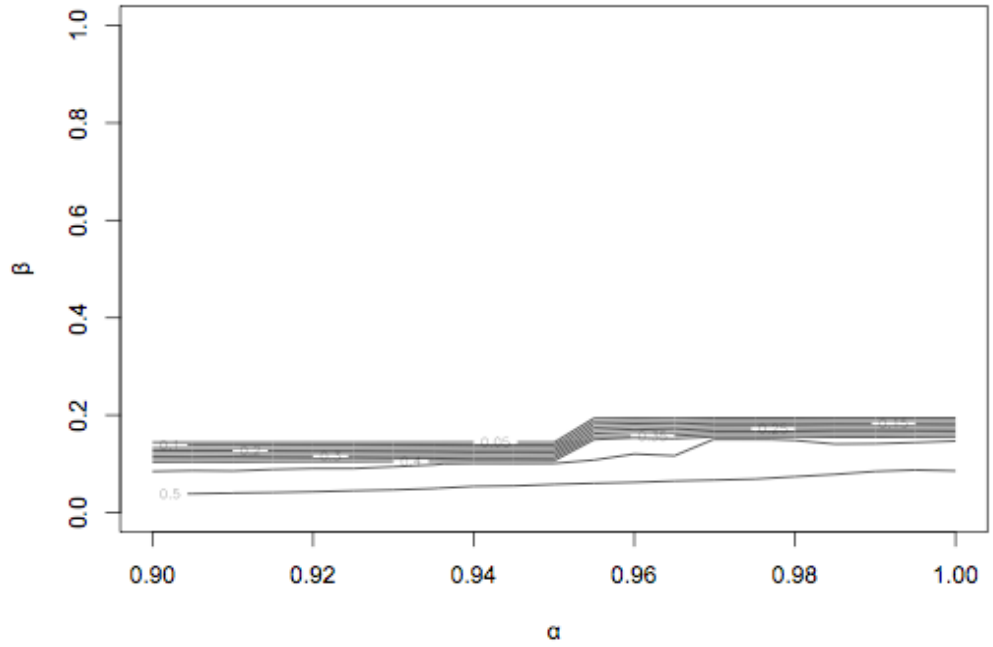
Det viser seg at vi oppnådde bedre resultater enn ved å bruke β=0,6. Vi kan finne en gjennomsnittlig β for de 100 treningssettene. Dette fremkommer av tabellen under.

Kovarians	Middel	St.avvik
Enkel	0,961	0,007
Multiplisert med 2	0,944	0,018

*Tabell 6: Middelverdi og standard avvik for β, når denne er estimert med pseudo-ML. Kovariansen er enten Σ eller 2*Σ.*

Dette skulle tilsi at dersom vi ønsker å finne den optimale β for å skille klassene i et bilde, så kan vi bruke denne metoden. Dette er gyldig uavhengig av om vi treffer alle klassene. Husk at for forsøkene med kovarians multiplisert med 2, så uteble klassen lilla fra resultatet. Likevel er β for de to forsøkene signifikante.

I mitt siste forsøk med de simulerte bildene ønsker jeg å se om bruk av kontekst på noen måte kan bidra til å skille klasser med stort overlapp. Resultatene så langt har ikke tydet på det, men ved å bruke metoden i kapittel 5.6 kan vi se hvilken kombinasjon av α og β som kan bidra til å opprettholde klassen lilla. Vi lager et treningssett med 5 % av pikslene. Deretter gjennomfører vi klassifiseringen gjentatte ganger med varierende α og β.



Figur 42: Plottet viser S for lilla gitt varierende α og β. Vi ser at økende β gir dårligere resultat, og at α har liten påvirkning.

Vi ser at dette ikke kan bidra til å forbedre resultatet. Maksimal S får vi for α=0,995 og β=0. Dermed kan det synes som at vi kan fastslå at bruk av kontekstuell klassifisering ikke bidrar til å skille klasser som har stort overlapp.

7 Utvelgelse av klasser

I kapittel 4 og 5 brukte jeg et fast treningssett. Dette utgjorde hele 41% av bildet, så det må sies å være et ganske omfattende treningssett. Det var delt inn i fem klasser, noe som virker litt merkelig når utgangspunktet vårt er å kartlegge siv. I tillegg kan valg av klasser virke litt tilfeldig. Blant annet har vi en klasse skog, som gjerne kunne vært delt inn i løvskog, eng og barskog. Så lenge vi opererer med siv som en egen klasse burde disse vært representert som egne klasser fra et naturfaglig perspektiv. Jeg vil i dette kapittelet forsøke å belyse hvilke kriterier man kan følge når man markerer treningssett for bruk i en tilsvarende kartleggingsoppgave.

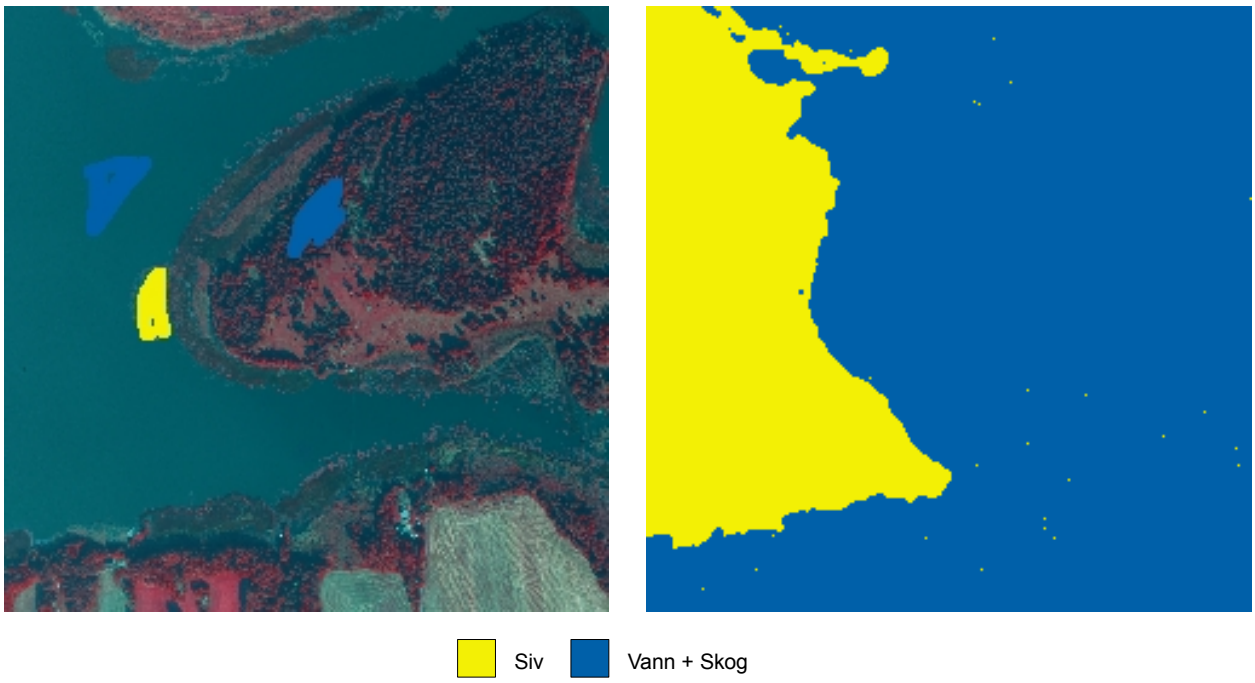
I dette kapittelet brukes samme bilde som i kapittel 4 og 5. I tillegg vil jeg kjøre med samme modell hele tiden. Dette er en kontekstuell modell med $\beta=1,5$, og parameterene for de multivariable normalfordelingene hentes direkte fra treningssettet.

For hver kjøring vil jeg presentere det treningssettet som er benyttet og det resulterende klassifiseringsresultatet. I tillegg vil jeg vise en forvirringsmatrise, som angir forskjellene mellom klassifiseringsresultatet og treningssettet fra kapittel 4 og 5.

Disse treningssettene er bygd opp uavhengig av hverandre, så vi bør kunne ha større tiltro til disse forvirringsmatrisene enn i kapittel 4 og 5. En ny problemstilling er at vi i dette kapittelet bruker treningssett med klasser som representerer sammenstillinger av de klassene jeg jobbet med tidligere. I disse tilfellene vil det ikke være så enkelt å gjøre noen direkte sammenstilling, men formålet med klassifiseringen er hele tiden siv og den er fast.

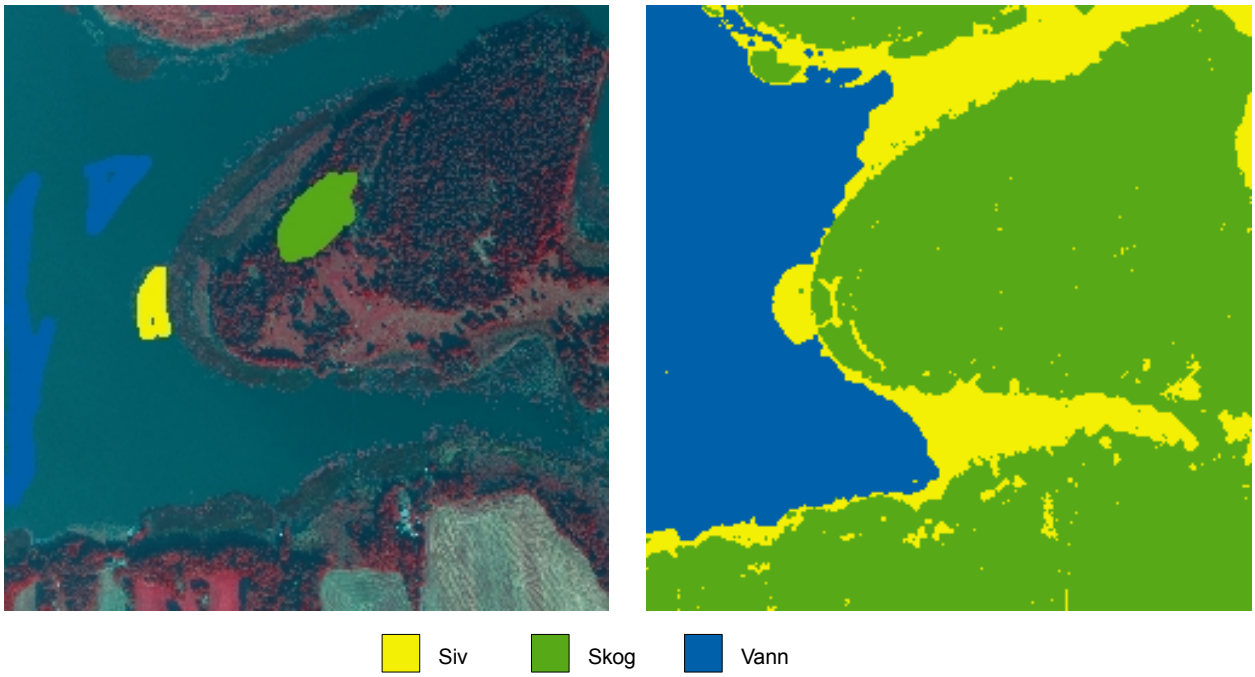
7.1 Ulike treningssett og resultater

Siden utgangspunktet for klassifiseringen har vært å dele opp i siv og ikke-siv, vil det være naturlig å tenke seg at man kun trenger to klasser. Derfor vil jeg i dette første eksempelet presentere hva som skjer dersom vi velger et treningssett med to klasser. Det ene dekker piksler med siv, mens det andre dekker piksler fra de tidligere klassene vann og skog. Av figur 5 ser vi at disse har fargeverdier på hver sin side av siv.



Figur 43: Bildet t.v. viser treningssettet. Her er det kun to klasser. Blått dekker både vann og skog, og siv er markert med gul. I resultatet t.h. ser vi at gult dekker områder med siv og vann, mens blått dekker det øvrige.

I figuren over ser vi til venstre treningssettet som er brukt. Det er delt opp i 3 separate områder av bildet. Det gule dekker områder med siv, mens de to blå områdene dekker helholdsvis vann og skog. Av resultatet ser vi at siv klarer å klassifisere sine egne treningspiksler, så sensitiviteten i modellen er god, men den tar samtidig med seg alle pikslene fra det blå feltet til venstre. Altså er presisjonen mindre god. Totalt sett må dette sies å være en dårlig klassifisering, og jeg ser ingen grunn til å presentere forvirringsmatrisen.



Figur 44: Treningssett består av tre klasser. Gult for siv, blått for vann og grønt for skog. Resultatet viser at vi klarer å skille de to, men siv okkuperer ganske store områder av det som tidligere ble identifisert som vann.

Her har jeg endret det ene blå området til grønt. I tillegg har jeg utvidet området med blått. Resultatet av dette er at jeg får separert de tre klassene: siv, vann og skog. Rent umiddelbart synes det som at resultatet i forhold til treningssettet er veldig god. Under har jeg satt opp en forvirringsmatrise mellom dette resultatet og treningssettet fra de foregående kapitlene.

		Treningssett						
		Siv	Skog	Jordbruk	Vann	Myr	Sum	Pres.
Klassifisert	Siv	140	5	1	2615	8	2769	5.06%
	Skog	49	7319	2611	116	1097	11192	65.39%
	Jordbruk	0	0	0	0	0	0	NaN%
	Vann	0	0	0	11779	0	11779	100.00%
	Myr	0	0	0	0	0	0	NaN%
	Sum	189	7324	2612	14510	1105	25740	
	Sens.	74.07%	99.93%	0.00%	81.18%	0.00%		
						Treffpros.	74.74%	

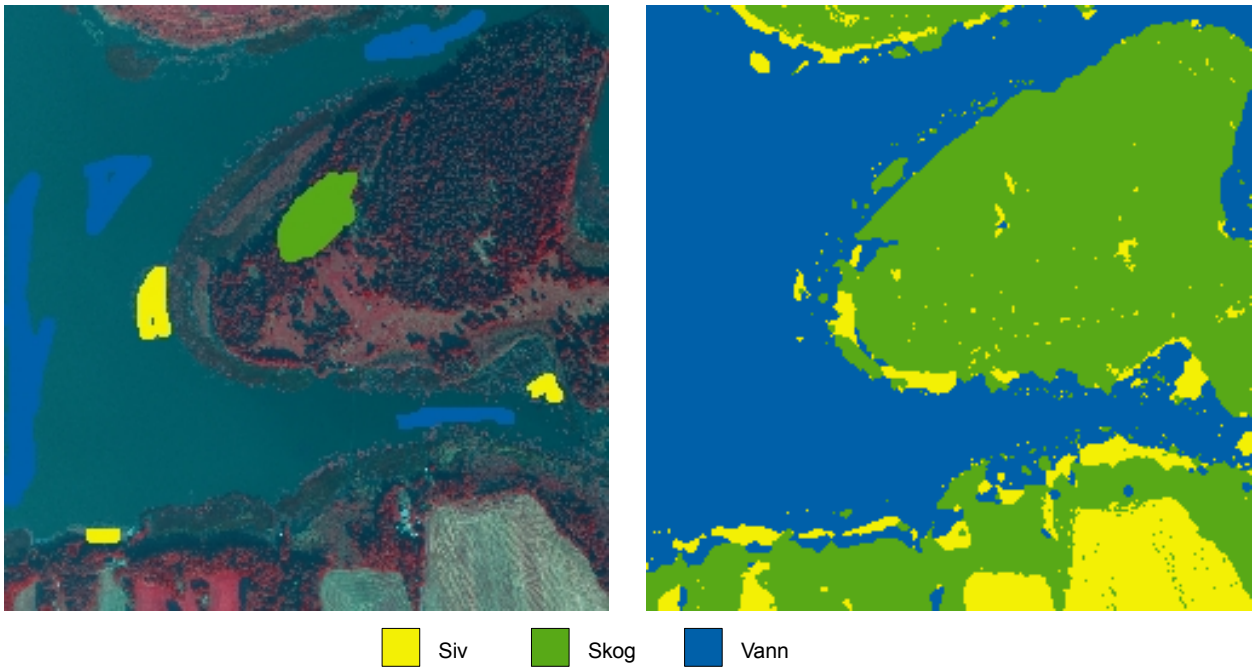
Figur 45: Forvirringsmatrise mellom klassifiseringsresultat og treningssett fra kapittel 4 og 5. Klassifiseringsresultatet inneholder kun klassene siv, skog og vann.

Rent umiddelbart virker en treffprosent på 74,7% forbausende godt. Dette skyldes nok at klassene vann og skog representerer store deler av bildet. Når vi klarer å skille godt mellom disse to klassene får vi en høy treffprosent. I tillegg ser vi at vann har en presisjon på 100%, samtidig som det har en

sensitivitet på 81%.

Resultatene for skog blir influert av at klassene for jordbruk og myr ikke er representert i dette treningssettet, men havner i klassen for skog. Dersom vi hadde satt opp en forvirringsmatrise som tok hensyn til dette. At skog var en sammenstilling av de opprinnelige klassene skog, jordbruk og myr, ville resultatet vært vel så godt som for vann.

Ser vi på resultatet for siv, ser vi at sensitiviteten er god mens presisjonen er mindre god. At vi har en dårlig presisjon skyldes at vi trekker til oss mange av pikslene som i treningssettet var vann. Dette ser vi godt på bildet. Over og under Feøya er det noen sund, som nå er dekket med siv. Tydeligvis er det en litt annen karakter på fargeverdiene her enn i de områdene jeg har markert som vann.



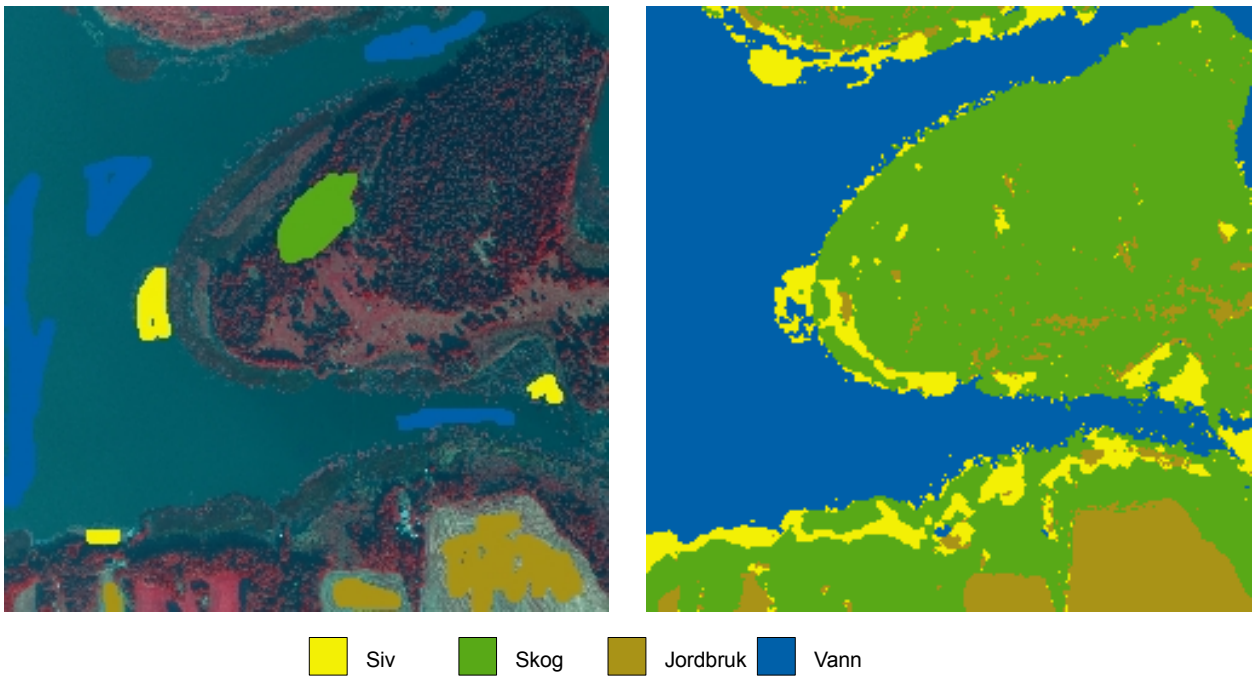
Figur 46: Flere felter i områder med vann gir bedre avgrensning for siv. Samtidig blir store områder inne på land nede til høyre klassifisert som siv. Tidligere har vi identifisert dette som jordbruksområder.

Ved å markere disse områdene med blått oppnår vi den ønskede effekten. De to sundene blir klassifisert som vann. Samtidig opplever vi at områder som var jordbruksområder i det opprinnelige treningssettet nå klassifiseres som siv. Dette skyldes at vi la til to områder med siv. Dersom fargeverdiene i disse to områdene lå nærmere jordbruksklassen enn skog, så vil disse områdene klassifiseres som siv. Dette ser vi også av forvirringsmatrisen på neste side.

Klassifisert		Treningssett						
		Siv	Skog	Jordbruk	Vann	Myr	Sum	Pres.
	Siv	87	15	2530	3	82	2717	3.20%
	Skog	0	7267	82	2	971	8322	87.32%
	Jordbruk	0	0	0	0	0	0	NaN%
	Vann	102	42	0	14505	52	14701	98.67%
	Myr	0	0	0	0	0	0	NaN%
	Sum	189	7324	2612	14510	1105	25740	
	Sens.	46.03%	99.22%	0.00%	99.97%	0.00%		
						Treffpros.	84.92%	

Figur 47: Forvirringsmatrise mellom klassifiseringsresultat og treningssett fra kapittel 4 og 5. Klassifiseringsresultatet inneholder kun klassene siv, skog og vann.

Den totale treffprosenten har økt. Det skyldes at sensitiviteten for vann har økt betraktelig. Nå er den på 99,9 %. Presisjonen for skog har også økt. Dette skyldes som vi så av bildet at siv har trukket til seg pikslene som lå i klassen jordbruk. I tillegg til at klassen siv har trekt til seg piksler fra jordbruk og myr, ser vi også at den har mistet endel piksler til vann. Både sensitiviteten og presisjonen for siv har gått ned.



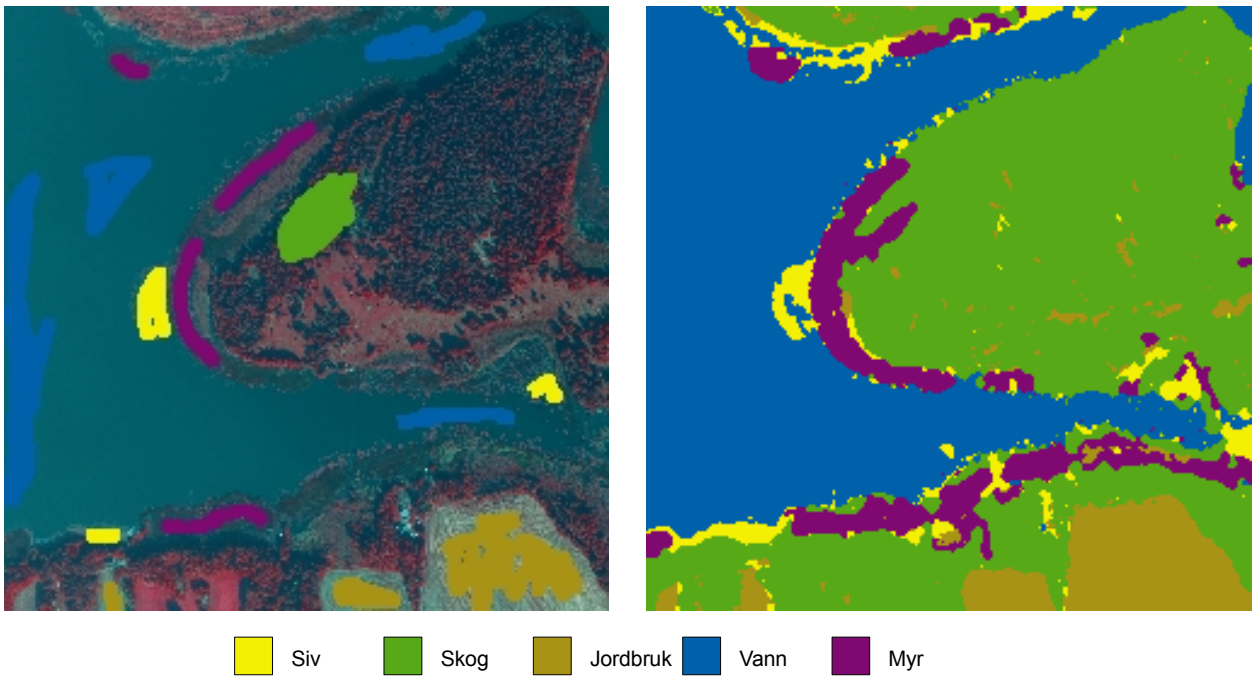
Figur 48: Ved å introdusere klassen jordbruk (oransje) i treningssettet t.v. oppnår vi at områdene i nedre kant ikke lenger markeres som siv.

Her har jeg forsøkt å kompensere for feilen med jordbruksområder, ved å definere disse som en egen klasse. Resultatet av dette synes umiddelbart å være veldig godt.

Klassifisert		Treningssett						
		Siv	Skog	Jordbruk	Vann	Myr	Sum	Pres.
	Siv	181	6	0	15	65	267	67.79%
	Skog	4	7301	19	10	968	8302	87.94%
	Jordbruk	0	17	2593	0	72	2682	96.68%
	Vann	4	0	0	14485	0	14489	99.97%
	Myr	0	0	0	0	0	0	NaN%
	Sum	189	7324	2612	14510	1105	25740	
	Sens.	95.77%	99.69%	99.27%	99.83%	0.00%		
						Treffpros.	95.42%	

Figur 49: Forvirringsmatrisen mellom klassifiseringsresultatet og treningssettet fra kapittel 4 og 5. Klassifiseringsresultatet inneholder ikke samme klasser som treningssettet. Vi har ikke med myr. Disse pikslene fordeler seg mellom siv, skog og jordbruk.

Dette hadde en god effekt på treffprosenten, som nå er oppe i 95,4 %. Alle klassene har nå fått en sensitivitet på over 95 %. Presisjonen er ikke like høy, men dette skyldes den manglende klassen myr. Pikslene som representerer myr i treningssettet havner i klassene siv, skog og jordbruk.
For å bedre presisjonen for siv, forsøker vi å legge til klassen myr. Under viser jeg det nye treningssettet sammen med et resultat av klassifiseringen.



Figur 50: T.v. treningssett med tilsvarende klasser som i kapittel 4 og 5, men ellers helt uavhengig. T.h. resultat av en klassifisering med β=1,5.

Det ser ut til at vi har fått plassert myr i riktige områder. På neste side har jeg satt inn

forvirringsmatrisen.

		Treningssett						
		Siv	Skog	Jordbruk	Vann	Myr	Sum	Pres.
Klassifisert	Siv	165	5	0	25	4	199	82.91%
	Skog	0	7297	4	5	882	8188	89.12%
	Jordbruk	0	22	2608	0	44	2674	97.53%
	Vann	0	0	0	14480	0	14480	100.00%
	Myr	24	0	0	0	175	199	87.94%
	Sum	189	7324	2612	14510	1105	25740	
	Sens.	87.30%	99.63%	99.85%	99.79%	15.84%		
						Treffpros.	96.06%	

Figur 51: Forvirringsmatrise mellom klassifiseringsresultat og treningssettet i kapittel 4 og 5. Vi oppnår en treffprosent på 96 %. Størst problem har vi med sensitiviteten til myr, som er på 15%. Dette skyldes at mange av de opprinnelige pikslene klassifiseres som skog.

I denne forvirringsmatrisen ser vi at vi har fått en forbedring i treffprosenten. I tillegg har vi fått en økt presisjon for siv, men sensitiviteten for siv har gått ned. Dette skyldes at enkelte av pikslene som var i klassen siv, nå blir klassifisert som myr.

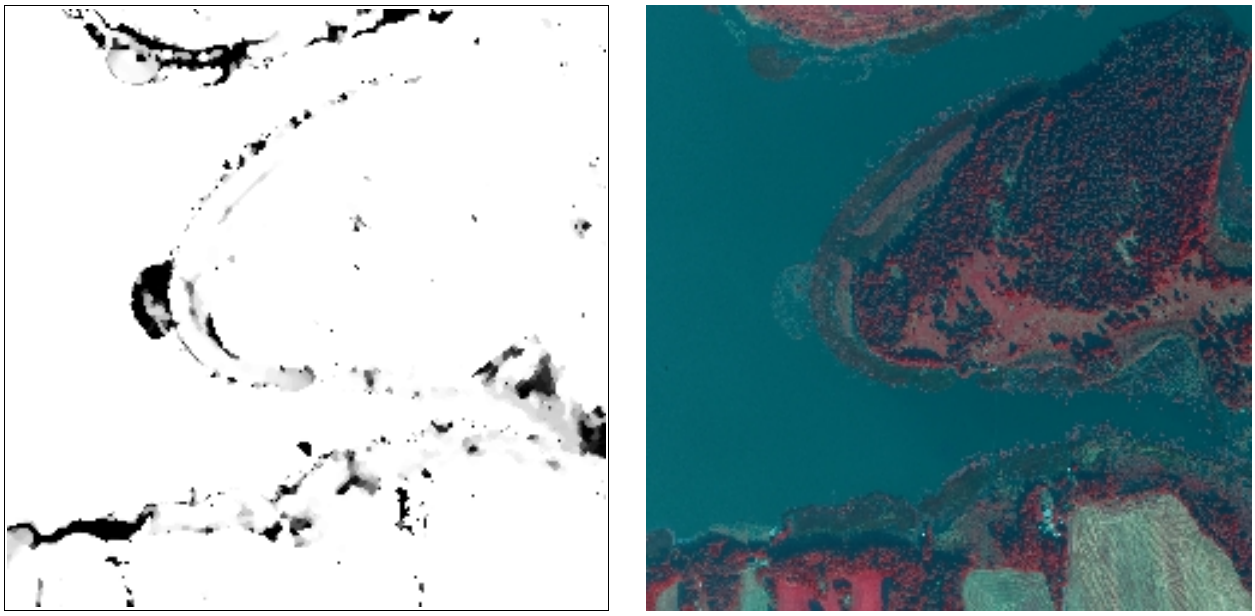
7.2 Gjentatte forsøk med klassifisering

Det kan virke som vi har funnet et tilfredsstillende sett med treningspiksler for å kartlegge siv. Samtidig har vi markert 11 % av bildet. I forrige kapittel så vi at det kan være tilstrekkelig å ha et treningssett på 5% av pikslene i bildet.

Derfor ønsker jeg å gjøre et forsøk med å gjenta klassifiseringen 100 ganger basert på at treningssettet er et uttrekk av pikslene i det treningssettet vi nå har bygd opp. Størrelsen på dette settet skal være slik at det utgjør 5 % av hele bildet, men likevel så stort at alle klasser blir representert med et visst antall piksler.

Det jeg ønsker med dette forsøket er å se om vi kan differensiere resultatet fra det siste forsøket og si at i enkelte områder er det større sannsynlighet for å finne siv enn i andre.

Resultatet av dette forsøket er vist i figuren på neste side.



Figur 52: Bildet til venstre viser hvor det er størst sannsynlighet for å finne siv. Sortere farge innebærer høyere sannsynlighet. T.h. har jeg lagt inn originalbildet, for å lettere kunne sammenligne.

Her ser vi resultatet av et mulig scenario. I bildet til venstre har vi kun fokus på siv. Områder som modellen har klassifisert som siv er markert med sort. Intensiteten i sortfargen indikerer hvor sannsynlig det er å finne siv i disse områdene. Dette er basert på at disse områdene ofte ender opp som siv i klassifiseringen.

Ved å sammenligne med originalbildet til høyre kan vi se at vi til en stor grad klarer å treffe vannkanten. Det ser også ut til at vi klarer å skille mellom det som ble betegnet som myr og siv. De områdene som havner inne på land, viser seg etter nærmere undersøkelser å være et hus med hage og en hyttetomt.

8 Overlappende bilder

Når vi bruker en statistisk metode for å analysere flyfotografiene er et av motivene å begrense den manuelle innsatsen. Dette vil være vel så viktig når vi har et sett med flyfotografier tatt opp over et større område på samme dag. Da vil det være nærliggende at det er en sammenheng mellom fargeverdiene i det ene bildet, som kan overføres til et nabobilde.

I tidligere kapitler har vi sett at vi er avhengig av en manuell innsats for å bygge opp et treningssett. Vi har ikke funnet noen god metode for å gjøre en automatisk klassifisering. I dette kapittelet skal vi se på om parameterene som er estimert på bakgrunn av treningssettet i et bilde også er brukbare i et overlappende bilde.

Som nevnt i kapittel 2 vil man i forbindelse med opptak av flyfotografier alltid legge inn et overlapp mellom bildene. Flyet vil som regel legge opp en plan med parallelle striper når de skal dekke et område. Langsetter stripene vil tidsdifferansen mellom bildeopptakene være kort, og man kan tenke seg liten variasjon i værforholdene. Vinkelen til enkelte objekter vil være forskjellig og dette kan ha betydning for skyggevirkninger. Når flyet kommer tilbake til et område i neste stripe vil det komme i motsatt retning. Også her vil vi ha en annen vinkel, men i dette bildet kan vi også få en større tidsdifferanse, som kan ha betydning for lysforholdene.

Et annet forhold er at bildene ikke er totalt overlappende. Et av bildene kan inneholde vegetasjons- eller landskapstyper som ikke finnes i det andre. Dersom dette ikke er med i treningssettet vil disse pikslene havne i en eller flere av de andre klassene. Dette trenger ikke være så farlig, men forteller oss at vi er avhengig av litt manuell innsats for hvert bilde i settet.

Figur 53: Uttrekk fra bilde 01398 i det området hvor det er overlapp med 01429. 01429 er bildet som ble brukt i kapittel 4 og 5. Det er mulig å kjenne igjen Feøya i nedre høyre hjørne.

Bildet vi brukte i kapittel 4 og 5 var merket 01429. Det overlapper med bildet merket med 01398, som jeg har lagt inn i figuren over. Det opprinnelige bildet vil jeg omtale som bilde A, mens bildet over vil jeg omtale som bilde B.

Bilde B er tatt opp i en parallell stripe, så flyet har kommet i motsatt retning av den det hadde ved opptak av bilde A. Derfor har jeg rotert det 180˚ for lettere å se sammenhengen med bilde A. I nedre høyre hjørne er det mulig å gjenkjenne Feøya.

Denne roteringen er ikke tilstrekkelig for å bruke bildene i en statistisk sammenheng. Det optimale er at det er et èn til èn forhold mellom hvert piksel. For å få til dette må vi gjennomføre en transformasjon tilsvarende den som er beskrevet i kapittel 2.2.

8.1 Geometrisk korreksjon

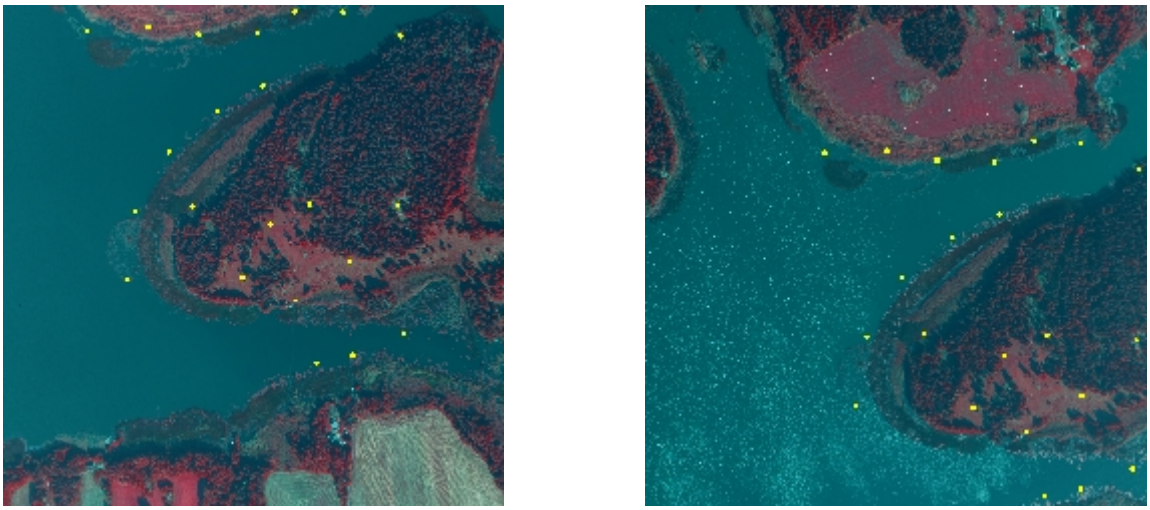
Det første vi må gjøre er å finne den affine transformasjonen som brukes mellom de to bildene.

$$x_B = a_1 + a_2 x_A + a_3 y_A$$

$$y_B = a_4 + a_5 x_A + a_6 y_A$$

For å bestemme parameterene $\{a_1 ... a_6\}$ plukker vi ut to sett med punkter i de to bildene som representerer samme geografiske lokalitet. Deretter bruker vi lineær regresjon på settet av $\{x_A, y_A, x_B\}$ for å finne $\{a_1, a_2, a_3\}$. Så gjør vi tilsvarende på $\{x_A, y_A, y_B\}$ for å finne $\{a_4, a_5, a_6\}$. Siden vi her snakker om to bilder som er tatt fra fly i samme opptak over et flatt terreng, bør vi kunne forvente en lav feilmargin.

Under har jeg lagt inn en figur som viser de to bildene, og settet med punkter som brukes for å komme frem til transformasjonen. Disse er markert med gule prikker.



Figur 54: Bildene med markering i gult for kontrollpunkter. Disse er plassert på samme geografiske lokalitet i de to bildene. Pikselet midt i hver markering brukes i en lineær regresjon for å komme frem til en affin transformasjon.

Med lineær regresjon får vi transformasjonen:

$$x_B = 759,692 + 1,006x_A - 0,031y_A$$

$$y_B = 950,814 + 0,033x_A + 1,008y_A$$

Begge regresjonene har en r^2 = 0,999, så her må vi kunne forvente bra overlapp mellom bildene.

Den inverse transformasjonen følger direkte av likningene over:

$$x_A = \frac{a_3a_4 - a_1a_6 + a_6x_B - a_3y_B}{a_2a_6 - a_3a_5} = -783.438 + 0.993x_B + 0.031y_B$$

$$y_A = \frac{a_1a_5 - a_2a_4 - a_5x_B + a_2y_B}{a_2a_6 - a_3a_5} = -917,620 - 0,033x_B + 0,991y_B$$

Nå ønsker vi å finne det rektangelet hvor det er overlapp mellom bildene. Først finner vi bilde B sine ytterpunkter uttrykt i koordinatsystemet til A. Begge bildene har en oppløsning på 4000 * 4000 piksler, så her bruker vi den inverse transformasjonen:

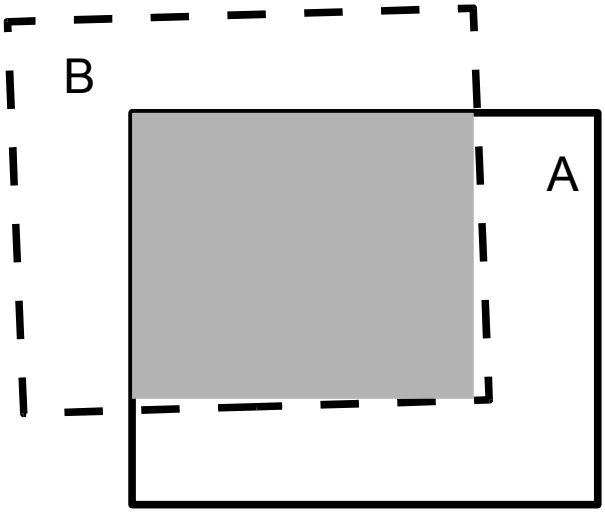
$$[x_B, y_B] = [0, 0] \Longleftrightarrow [x_A, y_A] = [-783, -918]$$

$$[x_B, y_B] = [0, 4000] \Longleftrightarrow [x_A, y_A] = [-659, 3046]$$

$$[x_B, y_B] = [4000, 0] \Longleftrightarrow [x_A, y_A] = [3189, -1050]$$

$$[x_B, y_B] = [4000, 4000] \Longleftrightarrow [x_A, y_B] = [3313, 2914]$$

Av dette ser vi at bilde B sitt nedre høyre hjørne ligger inne i bilde A, mens de øvrige hjørnene ligger utenfor. Samtidig ser vi at høyre side ligger høyere enn venstre. I figuren på neste side har jeg forsøkt å skissere hvordan bilde A og B ligger i forhold til hverandre.



Figur 55: Her ser vi omriss av de to bildene, og en tilnærmelse av geometrien gitt av transformasjonen. Det grå området viser overlappet mellom bildene, som vil være hovedfokus i den videre fremstillingen.

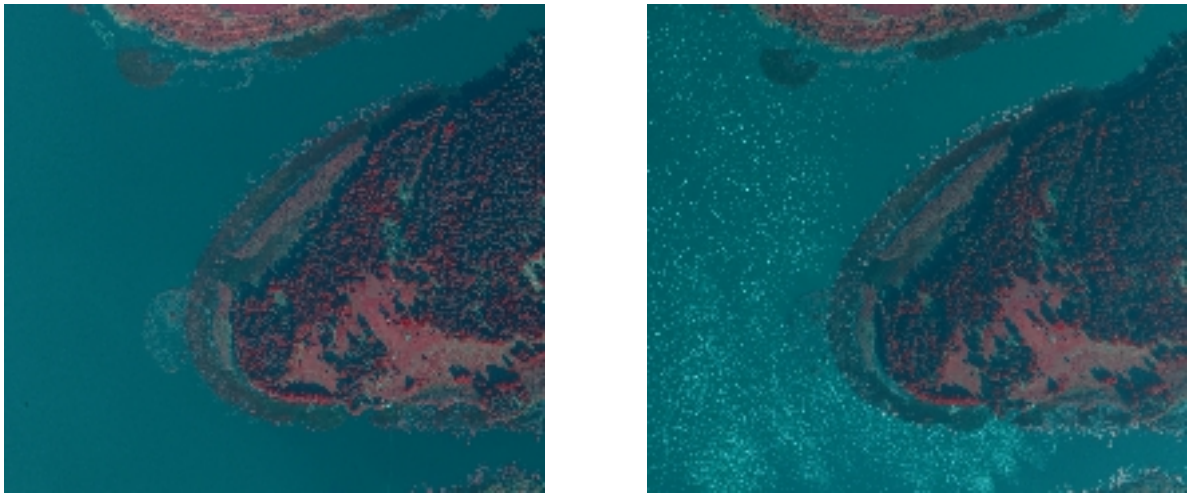
Vi kan fastslå at et rektangel som omslutter begge bildene må ha koordinatene

[-783, -1050, 4000, 4000].

For å finne overlappet mellom de to bildene, som er skravert med grått i figuren over, må vi finne skjæringspunktet mellom bilde B sin høyre kant og topplinja til bilde A. Av dette finner vi at rektangelet

[0,0,3222,2914]

har piksler i begge bildene. Fremdeles vil vi jobbe med en nedskalert versjon av bildet i analysene.



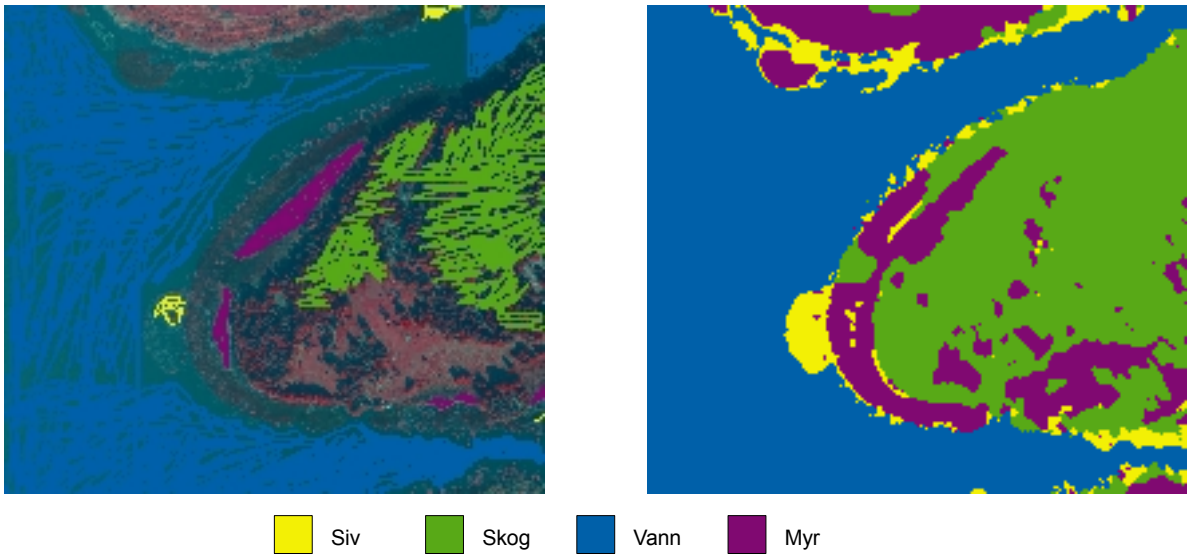
Figur 56: Bilde B (t.h.) er transformert til samme koordinatsystem som bilde A (t.v.) med en affin transformasjon. Begge bildene er beskåret slik at det er overlapp i alle piksler.

Etter en transformasjon av bilde B er bildene som vist i figuren over. Begge bildene er beskåret til rektangelet med fullt overlapp. Her ser vi at det er en klar likhet mellom bildene. Bilde B har mer hvitskimmer i vannet enn bilde A. Dette kan komme av solvinkelen, eller at det har blåst opp. Vi ser

også at det er litt mindre kontast i områdene med skog på bilde B.

8.2 Klassifisering

De gode resultatene fra regresjonen gir oss grunn til å tro at markeringen vi gjorde i bilde A kan brukes direkte i den transformerte utgaven av bilde B. Derfor vil vi igjen ta i bruk treningssettet fra figur 5. Det viser seg at klassen jordbruk ikke er representert i vårt utsnitt, så den holder vi utenom. Først gjennomfører vi en klassifisering i bilde A, og prøver å bruke samme parametersett i en klassifisering av bilde B. Gjennom dette ønsker vi å se om det er mulig å bruke et felles parametersett for flere bilder.



Figur 57: Treningssettet (t.v.) og klassifiseringsresultatet (t.h.) i bilde A. Her har vi kun brukt området med overlapp mllom de to bildene. Klassen jordbruk er ikke representert.

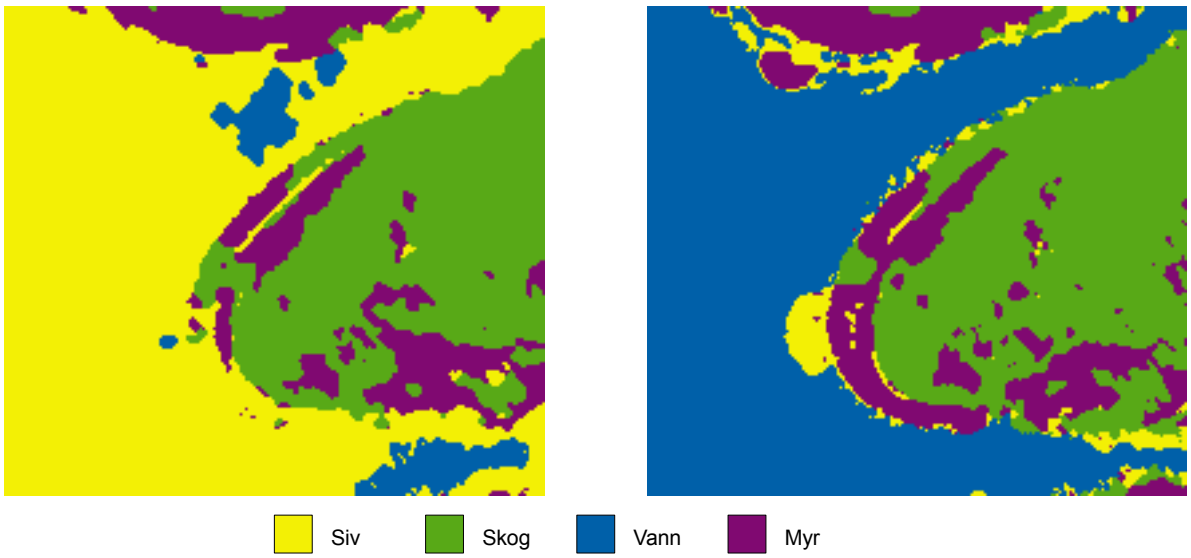
I første klassifisering har jeg brukt β = 2,0.

		Treningssett				
	Siv	Skog	Vann	Myr	Sum	Pres.
Klassifisert Siv	91	0	82	2	175	52.00%
Skog	0	3069	6	24	3099	99.03%
Vann	0	0	11633	0	11633	100.00%
Myr	4	14	0	530	548	96.72%
Sum	95	3083	11721	556	15455	
Sens.	95.79%	99.55%	99.25%	95.32%		
				Treffpros.	99.15%	

Figur 58: Forvirringsmatrise som viser resultatet av klassifisering innenfor området med overlapp mellom de to bildene. Presisjonen for siv er svak. Dette kan tilskrives at vi bruker β=2,0.

Resultatet ser veldig lovende ut, og forvirringsmatrisen viser at vi treffer ganske godt. Presisjonen for siv er ikke så veldig god, men dette kan skyldes at vi bruker så høy β.

På bakgrunn av denne klassifiseringen har vi fått et parametersett π_k, μ_k, Σ_k . I neste omgang ønsker jeg å bruke dette på bilde B uten å gjøre tilpasninger. Fordelingen mellom klassene π_k skal i prinsippet være den samme, så denne vil jeg også holde fast. Resultatet av denne klassifiseringen vises i figuren under.

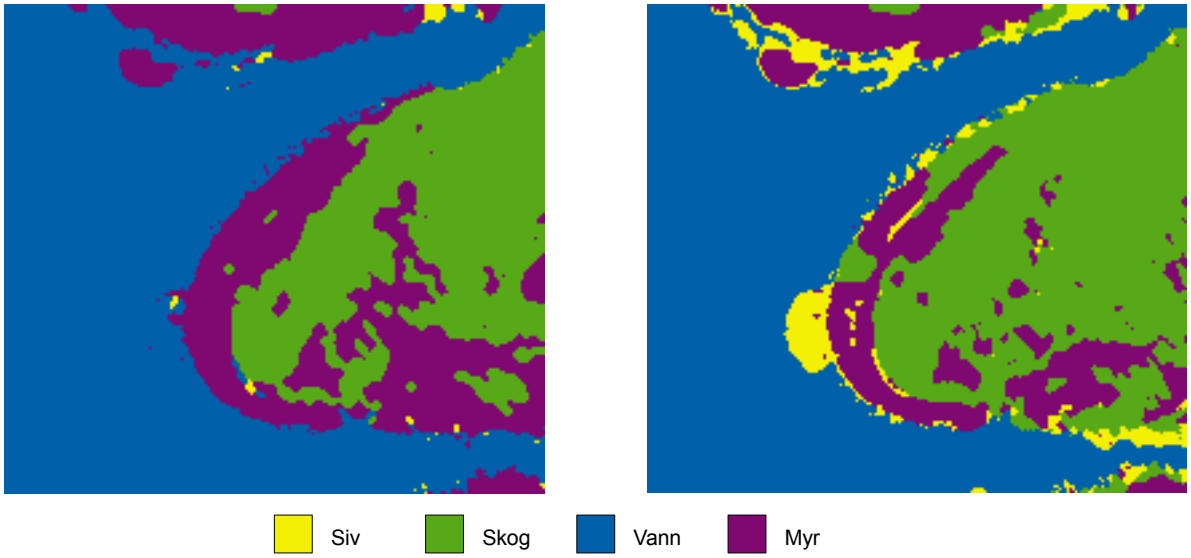


Figur 59: Resultat av å klassifisere bilde B med parametersett estimert i bilde A. T.v. vises resultatet i bilde B, mens t.h. vises resultatet i bilde A. Vi kan kjenne igjen noe av strukturen i bildet, men klassene siv og vann blandes sammen.

Her ser vi at siv er overrepresentert i områder med vann. Dette skyldes nok skinnet i vannet som jeg omtalte i tidligere.

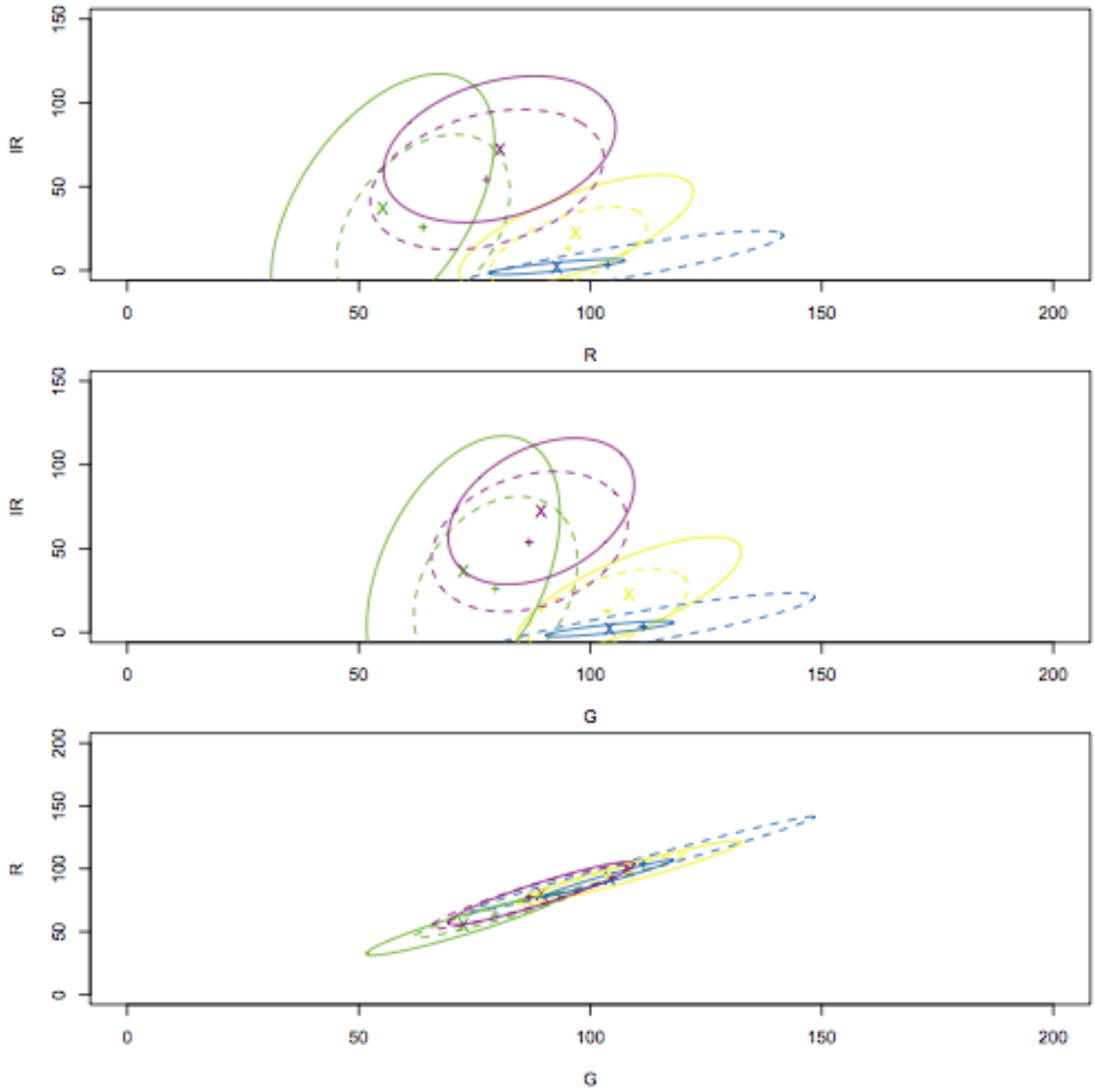
I neste omgang ønsker jeg å bruke lokaliseringen av treningspikslene i bilde A til å lage treningssett i bilde B. Dette kunne tilsvare at vi har et georeferert bilde, og så er vi ute i felten og markerer områder med ulike vegetasjonstyper som treningssett.

Det skal bortimot være et èn-èn forhold mellom pikslene i bilde A og B. Treningssettet ble bygd opp ved å markere områder i bilde A som var fremtredende. Disse trenger ikke nødvendigvis være det i bilde B. Resultatet vises i figuren på neste side.



Figur 60: Resultat av å bruke lokalisering av treningspiksler fra bilde A til å bygge opp treningssett i bilde B. T.v. resultatet i bilde B, og t.h. resultatet i bilde A.

Her ser vi at vannet kommer til sin rett fordi vi fanger opp endringen i fargeverdi denne har fra det ene opptaket til det neste. Samtidig ser vi at myrområdene får en mer fremtredende plass, mens siv har blitt fortrengt. Riktignok har vi noen flekker langs kanten av vannet, så plasseringen av disse synes å være riktig. I figuren på neste side har jeg lagt inn grafer som viser sannsynlighetsmassen for de ulike klassene, og hvordan de varierer mellom bildene.



Figur 61: Graf som viser de ulike parametersettenes sannsynlighetsmasse. Heltrukne linjer og X er basert på klassene estimert på bakgrunn av fargeverdiene i bilde A, mens stiplete linjer og + representerer klassene estimert på bakgrunn av fargeverdiene i bilde B.

Av figuren over ser vi at vi har fått en mye større variasjon i klassen vann, representert med blått. Samtidig som de andre klassene har gått mot lavere verdier, og mindre variasjon. Dette svarer til iakttagelsen av at skog hadde fått mindre kontrast. Vi ser også at vi har fått klarere overlapp mellom klassene myr og skog.

9 Sammendrag og videre utvikling

Hovedformålet med denne oppgaven har vært å se på bruken av fjernanalyse for å kartlegge vegetasjonen i en innsjø. Mer spesifikt har vi sett på bilder tatt opp fra fly over et område ved Vansjø i Østfold.

I kapittel 4 brukte jeg en ordinær klassifiseringsrutine hvor vi antok at pikslene i bildet kom fra hver sin klasse. Videre antok vi at hver av klassene hadde en multivariabel normalfordeling for å karakterisere pikslenes fargeverdier. Dermed kunne vi tilordne hvert piksel til den klassen med størst sannsynlighet gitt fargeverdien. Det viste seg at resultatene ble bra dersom vi estimerte parameterene direkte fra treningssettet, men at det var vanskeligere å finne en automatisert rutine. Vi forsøkte med en EM tilnærming, men den endte opp med å utelate klasser. Dette skyldes antakeligvis at det blir stort overlapp i forventingsfunksjonene til klassene.

Overlapp mellom klassene var også en problemstilling når parameterene ble estimert fra treningssettet. Dette klarte vi å kompensere for ved å benytte en regularisert kovariansmatrise. Ved tilpasningen av parameterene for denne brukte vi koeffisienter som fokuserte på feilklassifiseringen av siv. Dette fungerte godt, men vi fant ingen god rutine for å komme frem til optimale parametere.

Resultatene våre ble sammenlignet med det opprinnelige treningssettet, og vi kan ikke legge så mye vekt på disse. Det gjennomgående problemet med disse resultatene er at det er mye støy i bildet som dannes. Dermed er det vanskelig å se at dette skal være et godt grunnlag for å lage et kart over vegetasjonen i området.

Et botemiddel for dette kikket vi på i kapittel 5, ved å introdusere en kontekstuell sannsynlighet i klassifiseringen. Dette sørger for å fjerne støy i bildet, og vi får mer utglatting av grensene. Samtidig får vi en forbedring av resultatet.

Også her prøvde vi oss med en automatisk tilpasning av parametersettet. Nå består dette av parameterene i de multivariable normalfordelingene og parameteren β. Det viste seg at vi hadde de samme problemene som i kapittel 4 med at klassene gikk inn i hverandre. Vi gjorde også noen forsøk med kun å tilpasse parameteren β. Dette ga ikke noen umiddelbar effekt for klassifisering av siv. Det kunne virke som at parameteren vel så mye burde settes på bakgrunn av formålet med kartelggingen, og i hvilken grad vi ønsker å «glatte ut» områdene.

På den annen side så vi i kapittel 6 et eksempel på at denne parameteren kan trekkes ut av bildet. Her brukte vi to bilder som var simulert utfra den samme fasiten, men med forskjellige multivariable normalfordelinger. Likevel viste det seg at en automatisk tilpasning av parameteren β ga tilnærmet samme resultat. I samme kapittel så vi også at bruken av kontekst gir en forbedring av

klassifiseringsresultatet, men at det ikke bidrar til å skille klasser som har et stort overlapp i forventningsfunksjonene.

Dette gjenstår som den store utfordringen. Vi kikket nærmere på de i kapittel 7. Her bygget vi opp et treningssett stegvis, og for hvert trinn så vi på hvilken implikasjon utvidelsen hadde for sluttresultatet. Dette kan være et eksempel på praktisk bruk av løsningen. Det viste seg at vi fikk bedre og bedre resultat jo flere klasser vi la til. Vi så også at det er et poeng at de områdene vi inkluderer i en klasse er mest mulig homogene og skiller seg fra de andre klassene.

En liten indikasjon på dette fikk vi i kapittel 8. Her tok vi parametersettet vårt over til et nabobilde, som var helt overlappende med det opprinnelige bildet. Vi skulle tro at fargeverdiene i de ulike klassene var nogenlunde like, men det viste seg ikke være tilfelle. Resultatet vi så tilsier at vi er avhengig av å bygge opp et eget sett med klasser og treningssett for disse i hvert enkelt bilde individuelt.

9.1 Videre arbeid

Vi skal ikke se bort fra at resultatene av klassifiseringene ikke ville sett så gode ut dersom testsettet hadde vært basert på feltarbeider. Dette kunne vært gjennomført ved å gå ut i felten og markere utstrekningen for ulike vegetasjonstyper på et kart. Deretter oppretter man en geometrisk transformasjon av bildet til kartet og plukker ut de pikslene som ligger innenfor markeringene. Disse utgjør da treningssett for sine respektive klasser.

Sawaya et al. (2003) viser til arbeider hvor de har hatt med seg et georeferert satellittbilde ut i båt på innsjøen. Ved hjelp av GPS har de hatt kontinuerlig oppdatering av sin posisjon i bildet, og kunnet se på bildet samtidig med at de legger inn sine observasjoner av vegetasjonen. På denne måten er det mulig å legge opp til en mye mer differensiert vegetasjonskartlegging. Fremfor å bare se på siv og fravær av siv, kan vi skille mellom ulike sivarter.

Når man legger opp til et slikt arbeid er man avhengig av at bildet som hentes ned fra satellitten blir prosessert og georeferert kort tid i forkant av at man reiser ut i båt. Det ligger også feilkilder i posisjoneringen man må ta hensyn til. Dette gjelder både for geoprosseseringen og GPS-målingene.

Når man først har georeferert bildet er det naturlig å tenke at man kan bruke et kart over innsjøen som en avgrensning av sivvegetasjonen. Her ligger det en usikkerhet i nøyaktigheten til inntegningen av vannkanten. Ofte er sivvegetasjonen begrenset til tynne striper langs vannkanten, og dersom denne er tegnet inn litt feil vil det få uheldige konsekvenser. Solberg et al. (1996) hadde en tilsvarende problemstilling. De kombinerte fargeverdier fra satellittbilder med kartinformasjon om jordbruksområder til å kartlegge høstpløying i landbruket. En tilsvarende løsning kunne vært brukt i forbindelse med våre bilder.

En annen retning det er interessant å forfølge er å se på ulike oppløsninger. I denne oppgaven har jeg brukt et uttrekk av originalbildet. Dette har jeg gjort av den enkle grunn at hele bildet er så omfattende at det krever for mye regnekapasitet. Originalbildet har 15000*7500 piksler, som svarer til en oppløsnig på 20*20 cm. Dette er i overkant av hva vi trenger. I denne oppgaven gjorde jeg et uttrekk fra denne datamengden på hver 16. piksel.

Vi skal ikke se bort fra at vi ville oppnådd enda bedre resultat ved å gå opp et nivå i oppløsning. La oss si vi plukker hver 8. piksel fra det originale bildet. En slik løsning beskriver Comer et al. (1999). De bygde opp en MRF med flere oppløsninger samtidig. Dette gjorde de ved å se på oppløsningene som en pyramide. For hvert piksel i et overliggende lag var det 4 piksler i laget under. MRF-en ble bygd opp sånn at den tok hensyn til klassetilhørigheten for nabopikslene i

samme oppløsning, og klassetilhørigheten for pikselet i den overliggende oppløsningen.

En alternativ løsning kan være å bygge opp en form for beslutningstre. Dette innebærer at vi begynner øverst i pyramiden med en klassifisering slik jeg har gjort i denne fremstillingen. For alle piksler som har klassetilhørighet over et gitt nivå bruker vi denne for alle tilhørende piksler i underliggende lag. For de pikslene som lå under grensen går vi til nivået under, og gjennomfører en ny klassifisering i dette nivået.

I en slik løsning må vi anta at alle parameterene utenom β er den samme for alle nivåene i pyramiden. Parameteren β vil man være nødt til å sette, og man må også bestemme nivået for når man skal akseptere en klassetilordning.

Et annet forhold som gjør en slik tilnærming attraktiv er at programvare som Google Earth og NASA World Wind bygger på at satellittbildene skaleres ved å plukke hvert andre piksel i overgangen fra ett nivå til det øvre. Dermed kan man se for seg en løsning hvor bildet blir klassifisert på ytterste nivå. Etterhvert som brukeren zoomer inn i bildet klassifiseres nye bilder med stadig høyere oppløsning.

Det vi også har sett er at treningssettene vi utarbeider ikke nødvendigvis trenger å være så omfattende. Når en person markerer i bildet vil han automatisk dekke mange piksler, som gir stabile estimater for klassenes parametersett. Faktisk er omfanget av treningssettet så omfattende at det kan tenkes man kan utnytte det for å gi et estimat på hvor godt resultatet vårt er. Dette kan vi gjøre ved å gjennomføre gjentatte uttrekk av treningssett for deretter å gjennomføre en klassifisering. I kapittel 7.2 ga jeg et eksempel på en slik fremgangsmåte.

I kapittel 8 så vi på bruk av samme parametersett i flere bilder, og at dette ikke nødvendigvis lar seg kombinere. Dette kan man se nærmere på ved å se om det er generelle endringer i fargeverdiene fra det ene bildet til det neste. Kanskje bildet er tatt opp senere på dagen, og alle piksler har blitt litt lysere? En løsning på dette er å innføre en gruppering av klasser. Vi kunne tenkt oss at siv var en gruppe av klasser. En i skyggeområder og en annen i mer åpne områder. Dette kan løses ved å bruke andre fordelingsfunksjoner enn den multivariable normalfordelingen jeg har brukt i denne oppgaven.

Det er mye som kan gjøres, men det viktigste er nok å se på hvordan dette kan brukes til å kartlegge makrofytter og ikke siv.

Referanser

Banko G. (1998). A Review of Assessing the Accuracy of Classifications of Remotely Sensed Data and of Methods Including Remote Sensing Data in Forest Inventory. Interim report International Institute for Applied Systems Analysis.

Besag J. (1974). Spatial Interaction and the Statistical Analysis of Lattice Systems. Journal of the Royal Statistical Society. Series B, Vol. 36

Besag J. (1986). On the Statistical Analysis of Dirty Pictures. Journal of the Royal Statistical Society. Series B, Vol. 48, No. 3.

Celeux G., Govaert G. (1993). Gaussian Parsimonious Clustering Models. Pattern Recognition, Vol. 28, Issue 5.

Comer M. L. og Delp E. J. (1999). Segmentation of Textured Images Using a Multiresolution Gaussian Autoregressive Model. IEEE Transactions on Image Processiong. Vol 8, No. 3

Dubes R. C. og Jain A. K. (1989). Random field models in image analysis. Journal of Applied Statistics, Vol. 16, No. 2

Friedman J. (1989). Regularized Discriminant Analysis. Journal of the American Statistical Association, Vol. 84, No. 405.

Geman G. og Geman D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 6.

Griffith J. A., Stehman S. V., Sohi T. L. og Loveland T. R. (2003). Detecting trends in landscape pattern metrics over a 20-year period using a sampling-based monitoring programme. International Journal of Remote Sensing, Vol. 24, No. 1.

Hastie T., Tibshirani R. og Friedman J. (2001). The Elements of Statistical Learning.

Hjort, Nils Lid (1986). Notes on the Theory of Statistical Symbol Recognition. Norsk regnesentral report nr 778/1986.

Kato Z. og Pong T.-C. (2006). A Markov random field image segmentation model for color textured images. Image and Vision Computing 24

Lehmann A. og Lachavanne J.-B. (1997). Geographic information systems and remote sensing in aquatic botany. Aquatic Botany 58.

Lillesand T. M., Kiefer R. W. (1987). Remote Sensing and Image Interpretation. Second Edition.

Marroquin M., Mitter S., og Poggio T. (1987). Probabilistic solution of illposed problems in computational vision. Journal American Statistical Association, Vol. 82

Mjelde M., Berge D., Stabbetorp O. (2008). Strandvegetasjonen i Vansjø. Kartlegging og forvaltningsstrategi. NIVA rapport 5813-2009.

Noda H., Shirazi M. N., Nogawa T. og Kawaguchi E. (1996). Unsupervised segmentation of multispectral images using hierarchical MRF model. The 1996 IEEE Signal Processing Society Workshop, Kyoto, Japan

Rogan J., Franklin J. og Roberts D. A. (2001). A comparison of methods for monitoring multitemporal vegetation change using Thematic Mapper imagery. Remote Sensing of Environment 80.

Rørslett B. og Lydersen E. (1980). Vegetasjonskartlegging av Vansjø. NIVA prosjektrapport O-78053.

Sawaya K. E., Olmanson L. G., Heinert N. J., Brezonik P. L. og Bauer M. E. (2003). Extending satellite remote sensing to local scales: land and water resource monitoring using high-resulution imagery. Remote Sensing of Environment 88.

Schistad Solberg A. H., Taxt T. og Jain A. K. (1996). A Markov Random Field Model for Classification for Multisource Satellite Imagery. IEEE Transactions on GeoScience and Remote Sensing. Vol. 34, No. 1

Silva T. S. F., Costa M. P. F., Melack J. M. og Novo E. M. L. M. (2007). Remote sensing of aquatic vegetation: theory and applications. Environmental Monitoring and Assessment, Vol. 140, No. 1-3

Toutin T. (2007). Review article: geometric processing of remote sensing images: models, algorithms and methods. International Journal of Remote Sensing, Vol. 25, No. 10

Valta-Hulkkonen K., Pellikka P., Tanskanen H., Ustinov A. og Sandman O. (2003). Digital false colour aerial photographs for discrimination of aquatic macrophyte species. Aquatic Botany 75

R kildekode

Følgende biblioteker er brukt:

```
library(rJava)
.jinit(classpath="markov-chain-1.0.0.jar", parameters="-Xmx512m")
library(EBImage)
library(Matrix)
```

Java-biblioteket som er brukt: markov-chain-1.0.0.jar er tilgjengelig via Subversion hos NIVA:

svn://kvina.niva.no/MarkovChain

Under følger de funksjonene som er kjørt i R. Den mest sentrale heter kjorFullKlassifisering og ligger i dermed i midten. (Dessverre er ikke alt like godt kommentert)

```
# Exponential likelihood - brukes i input til nlm. Det er bakgrunnen for fortegnet.
# param - beta som skal estimeres
# count - Matrise med h(x) for hver beta.
#             Dersom det er flere h(x) enn beta-verdier, økes antall beta.
#             Dette svarer til situasjoner med felles beta
# total - Matrise med aktuelle h(i), hvor i er en aktuell klasse for denne beta.
#                         (Vi har luket ut de som vil gi 0)
# klasseInx - Angir hvilke kolonner i total og param som er aktuell for en gitt klasse.
#                                 (Inneholder tre kolonner klasse, inxBeta og inxTotal)
exponentialLikelihood <- function(param, count, total, klasseInx) {

     Cl = length(count[1,])
     if (Cl > length(param)) {
          beta = rep(param,Cl)
     }
     else {
          beta = param
     }


     N = length(count[,1])
     K = max(klasseInx[,1])
     u = array(N)
     t = matrix(0,nrow=N,ncol=K)
     u = count %*% beta

     for (j in 1:K) {
          i = which(klasseInx[,1]==j)
          if (length(i) > 1)
               t[,j] = exp(total[,klasseInx[i,3]] %*% beta[klasseInx[i,2]])
```

```
          else
               t[,j] = exp(total[,klasseInx[i,3]] %*% t(beta[klasseInx[i,2]]))
     }

     -sum(u - log(rowSums(t)))
}

# Klassifiserer populasjonen y etter en regularisert modell.
# alfa = 1 gir ingen regularisering, og sigma brukes ikke.
# alfa = 0 gir full regularisering, og sigma_k brukes ikke.
# sigma svarer til den globale kovariansen
# sigma_k gir kovariansen for hver klasse
# mu_k girmiddel for hver klasse
# pi_k gir fordeling mellom klassene
# Returnerer prediksjon som en array, sannsynligheten for denne og de øvrige.
regularizedDiscriminant <- function(alfa, sigma, sigma_k, pi_k, mu_k, y) {
     library(Matrix)
     K = length(pi_k)
     N = length(y[,1])
     d = length(y[1,])

     pred = rep(1,N)
     mp = array(dim=c(N,2))
     p = array(dim=c(N,K))
     c = 2*pi^d
     for (i in 1:K) {
          if (pi_k[i] > 0) {
               if (alfa==1) {
                    sig = sigma_k[i,,]
               }
               else if (alfa==0.0) {
                    sig = sigma
               }
               else {
                    sig = alfa*sigma_k[i,,]+(1-alfa)*sigma
               }

               mah = mahalanobis(y, mu_k[i,], sig)
               p[,i] = pi_k[i]*exp(-0.5*mah)/sqrt(c*norm(sig))
               mp[,2] = p[,i]
          }
          else {
               p[,i] = 0
               mp[,2] = 0
          }

          if (i == 1) {
               mp[,1]=mp[,2]
          }
          else {
               inx = which(mp[,2]>mp[,1])
               pred[inx]=i
               mp[inx,1]=mp[inx,2]
          }
     }

     list(prediction=pred,p0=mp[,1],p=p)
}
```

```
# Funksjon som gjennomfører en klassifisering.
# Enten innebærer dette å kalle regularizedDiscriminant,
# eller så kaller vi både denne og deretter en kontekstuell klassifisering.
classificationRound <- function(pop, mu_k, sigma_k, sigma, pi_k, jconv, Cl, alfa, beta,
imgSize) {
     P = length(pop[1,])
     N = length(pop[,1])
     K = length(mu_k[,1])
     n_k = array(dim=K)

     ret_max = regularizedDiscriminant(alfa, sigma, sigma_k, pi_k, mu_k, pop)
     p = ret_max$p / rowSums(ret_max$p)
     if (!is.null(jconv)) {
          pred = ret_max$prediction - 1
          beta_arr = rep(beta, Cl)

          ret_conv = .jcall(jconv,"[I","classify", as.double(p), as.integer(Cl),
as.integer(imgSize[1]), as.integer(imgSize[2]), as.double(beta_arr), as.integer(20))

          ret_conv=ret_conv+1

          list(prediction=ret_conv, p=p)
     }
     else {
          list(prediction=ret_max$prediction, p=p)
     }
}

kjorEnkelKlassifisering <- function(pop, train, K, imgSize, iter = 1) {

     N = length(pop[,1])
     d = length(pop[1,])
     ret_cross = array(0,dim=N)

     pred = array(dim=N)

     pi_k = array(dim=K)
     mu_k = array(dim=c(K,d))
     sigma_k = array(dim=c(K,d,d))
     n = sum(is.finite(train))

     for (j in 1:K) {
          i_k = which(train==j)
          pi_k[j] = length(i_k) / n
          mu_k[j,] = colMeans(pop[i_k,])
          sigma_k[j,,] = cov(pop[i_k,])
     }


     for (i in 1:iter) {
          print(cat("Iterasjon ",i))
          print(pi_k)
          ret_cross = classificationRound(pop, mu_k, sigma_k, sigma=NULL, pi_k,
jconv=NULL, Cl, alfa=1.0, beta=0.0, imgSize)
          pred = ret_cross$prediction

          if (i < iter) {
```

```
                #Estimere nye parametere

                for (k in 1:K) {
                     pi_k[k] = sum(pred==k)/N
                }
           }
     }

     list(prediction=pred, mu_k=mu_k, sigma_k=sigma_k, pi_k=pi_k, beta=beta)
}

# Hovedfunksjon for å gjennomføre alle klassifiseringer
# pop - populasjonen
# train - treningssettet som skal brukes til å bygge opp parametere. Kan settes til NULL
dersom pi_k, mu_k, sigma, sigma_k spesifiseres
# K - antall klasser
# imgSize - størrelsen på bildet
# iter - antall iterasjoner som skal gjennomføres
# alfa - alfaparameteren i regularisert klassifisering. Kan være en vektor.
# beta - beta-parameteren for kontekstuell klassifisering. Kan være en vektor.
# pi_k - brukes dersom train=NULL
# mu_k - brukes dersom train=NULL
# sigma - brukes dersom train=NULL
# sigma_k - brukes dersom train=NULL.
# adjustMuSigma - skal vi kjøre EM på mu og sigma
# adjustBeta - skal vi kjøre pseudo-ML på beta
# jconv - java-klasse for å håndtere kontekstuell klassifisering
kjorFullKlassifisering <- function(pop, train=NULL, K, imgSize, iter = 1, alfa = 1.0,
beta = 0.0, pi_k=NULL, mu_k=NULL, sigma=NULL, sigma_k=NULL, adjustMuSigma=FALSE,
adjustBeta=FALSE, jconv = NULL) {

     N = length(pop[,1])
     d = length(pop[1,])
     ret_cross = array(0,dim=N)


     if (is.null(jconv) & !(beta == 0.0 & adjustBeta == FALSE)) {
          jconv = .jnew("niva/stats/markov/r/Classifier")
          for (j in 1:K) {
               .jcall(jconv,,"clique",as.integer(j-1))
          }
          Cl = K
     }

     pred = array(dim=N)

     if (!is.null(train)) {
          pi_k = array(dim=K)
          mu_k = array(dim=c(K,d))
          sigma_k = array(dim=c(K,d,d))
          sigma = cov(pop[which(is.finite(train)),])
          n = sum(is.finite(train))

          for (j in 1:K) {
               i_k = which(train==j)
               pi_k[j] = length(i_k) / n
               mu_k[j,] = colMeans(pop[i_k,])
               sigma_k[j,,] = cov(pop[i_k,])
```

```
          }
     }

     for (i in 1:iter) {
          print(cat("Iterasjon ",i))
          print(pi_k)
          ret_cross = classificationRound(pop, mu_k, sigma_k, sigma, pi_k, jconv, Cl,
alfa, beta, imgSize)
          pred = ret_cross$prediction

          if (i < iter) {
               #Estimere nye parametere

               if (adjustBeta == TRUE) {
                    cnt = .jcall(jconv,"[I", "counts", as.integer(imgSize[1]),
as.integer(imgSize[2]))
                    cnt = matrix(cnt, ncol=Cl)

                    klInx = .jcall(jconv,"[I","totalIndex")
                    klInx = matrix(klInx, ncol=3)

                    tot = .jcall(jconv,"[I","total")
                    tot = matrix(tot,ncol=max(klInx[,3]))

                    ret_beta = nlm(exponentialLikelihood, beta, count=cnt, total=tot,
klasseInx=klInx)
                    beta = ret_beta$estimate
                    print(beta)
               }


               if (beta != 0.0) {
                    convProb = .jcall(jconv,"[D","probabilities",
as.integer(imgSize[1]), as.integer(imgSize[2]))
                    p = matrix(convProb, ncol=K)
               }
               else {
                    p = ret_cross$p
               }

               if (adjustMuSigma == TRUE) {

                    for (k in 1:K) {
                         resp = p[,k]/rowSums(p)
                         cv = cov.wt(pop,resp,method="ML")
                         mu_k[k,] = cv$center
                         sigma_k[k,,] = cv$cov

                         pi_k[k] = sum(resp)/N
                    }
               }
               else {
                    for (k in 1:K) {
                         resp = p[,k]/rowSums(p)
                         pi_k[k] = sum(resp)/N
                    }
               }
          }
```

```
	}

	list(prediction=pred, mu_k=mu_k, sigma=sigma, sigma_k=sigma_k, pi_k=pi_k, beta=beta)
}

# Bildene må ha samme dimensjon. Treningssettene har felles klasser, så K står for totalt
antall klasser.
kjorToBilderKlassifisering <- function(popA, popB, trainA, trainB, K, imgSize, iter = 1,
beta = 0.0, adjustMuSigma=FALSE, adjustBeta=FALSE, jconv = NULL) {

	N = length(popA[,1])
	d = length(popA[1,])
	ret_cross = array(0,dim=N)

	if (is.null(jconv) & !(beta == 0.0 & adjustBeta == FALSE)) {
		jconv = .jnew("niva/stats/markov/r/Classifier")
		for (j in 1:K) {
			.jcall(jconv,,"clique",as.integer(j-1))
		}
		Cl = K
	}

	pred = array(dim=N)

	pi_k = array(dim=K)
	mu_k = array(dim=c(K,d))
	sigma_k = array(dim=c(K,d,d))

	n = sum(is.finite(trainA)) + sum(is.finite(trainB))

	for (j in 1:K) {
		i_k_a = which(trainA==j)
		i_k_b = which(trainB==j)
		if (length(i_k_a) > 0) {
			pi_k[j] = length(i_k_a) / n
			mu_k[j,] = colMeans(popA[i_k_a,])
			sigma_k[j,,] = cov(popA[i_k_a,])
		}
		else {
			pi_k[j] = length(i_k_b) / n
			mu_k[j,] = colMeans(popB[i_k_b,])
			sigma_k[j,,] = cov(popB[i_k_b,])
		}
	}


	for (i in 1:iter) {
		print(cat("Iterasjon ",i))
		print(pi_k)
		ret_cross_a = classificationRound(popA, mu_k, sigma_k, NULL, pi_k, jconv, Cl,
1.0, beta, imgSize)
		pred_a = ret_cross_a$prediction
		if (i < iter) {
			if (beta != 0.0) {
				convProb = .jcall(jconv,"[D","probabilities",
as.integer(imgSize[1]), as.integer(imgSize[2]))
				p_a = matrix(convProb, ncol=K)
			}
```

```
                else {
                    p_a = ret_cross_a$p
                }
            }

            ret_cross_b = classificationRound(popB, mu_k, sigma_k, NULL, pi_k, jconv, Cl,
1.0, beta, imgSize)
            pred_b = ret_cross_b$prediction
            if (i < iter) {
                if (beta != 0.0) {
                    convProb = .jcall(jconv,"[D","probabilities",
as.integer(imgSize[1]), as.integer(imgSize[2]))
                    p_b = matrix(convProb, ncol=K)
                }
                else {
                    p_b = ret_cross_b$p
                }
            }

            if (i < iter) {
                #Estimere nye pi
                for (k in 1:K) {
                    sums = rowSums(p_a) + rowSums(p_b)
                    resp_a = p_a[,k]/sums
                    resp_b = p_b[,k]/sums
                    pi_k[k] = (sum(resp_a)+sum(resp_b))/(2*N)
                }
            }
        }

        list(predictionA=pred_a, predictionB=pred_b, mu_k=mu_k, sigma_k=sigma_k, pi_k=pi_k)
}


# Sensitivity - Evne til å få med seg alt
# Specificity - Evne til ikke å overdrive
countTreff <- function(test, prediction, class=1) {
        i_c = which(test==class)
        sens = sum(prediction[i_c] == class, na.rm=TRUE) / (length(i_c) + 1)
        prec = sum(prediction[i_c] == class, na.rm=TRUE) /
(sum(prediction[which(is.finite(test))] == class) + 1)

        list(sensitivity=sens, precition=prec)
}



kjorItereringAlfaBeta <- function(pop, train, K, imgSize, iter, alfa, beta, test=NULL,
adjustMuSigma=FALSE, class=1) {
        N = length(pop[,1])
        d = length(pop[1,])

        k = length(alfa)
        l = length(beta)

        ret = array(dim=c(k,l,2))

        if (l > 1 | beta != 0.0) {
```

```
          jconv = .jnew("niva/stats/markov/r/Classifier")
          for (j in 1:K) {
               .jcall(jconv,,"clique",as.integer(j-1))
          }
          Cl = K
     }
     else {
          jconv = NULL
     }


     for (j in 1:l) {
          print(beta[j])
          for (i in 1:k) {
               print(alfa[i])
               if (l>1) {
                    ret_class = kjorFullKlassifisering(pop, train, K, imgSize, iter,
alfa=alfa[i], beta=beta[j], adjustMuSigma=adjustMuSigma, adjustBeta=FALSE, jconv=jconv)
               }
               else {
                    if (is.null(jconv)) {
                         ret_class = kjorFullKlassifisering(pop, train, K, imgSize,
iter, alfa=alfa[i], beta=beta, adjustMuSigma=adjustMuSigma, adjustBeta=FALSE, jconv=NULL)
                    }
                    else {
                         ret_class = kjorFullKlassifisering(pop, train, K, imgSize,
iter, alfa=alfa[i], beta=beta, adjustMuSigma=adjustMuSigma, adjustBeta=TRUE, jconv=jconv)
                    }
               }


               if (is.null(test)) {
                    cnt = countTreff(train, ret_class$prediction, class)
               }
               else {
                    cnt = countTreff(test, ret_class$prediction, class)
               }


               ret[i,j,1] = cnt$sensitivity
               ret[i,j,2] = cnt$precition
          }
     }
     ret
}



# Oppretter java-komponent med fast oppsett av clique
# Kjører gjentatte classificationRound på denne
crossvalidation <- function(pop, classes, K, imgSize, iter, beta=0.0, fraction=0.0,
antall=0, adjustMuSigma=FALSE, adjustBeta=FALSE, iterRunde=10) {
     N = length(pop[,1])
     ret_cross = array(0,dim=N)
     train = array(dim=N)
     betaArr = beta
     if (fraction > 0.0) {
          n = length(which(is.finite(classes)))*fraction
```

```
	}
	else {
		n = K*antall
	}
	i_train = array(dim=n)

	if (!beta==0.0 | adjustBeta==TRUE) {
		jconv = .jnew("niva/stats/markov/r/Classifier")
		for (i in 1:K) {
			.jcall(jconv,,"clique",as.integer(i-1))	}
	}
	else {
		jconv = NULL
	}

	if (adjustBeta==TRUE) {
		betaArr = array(dim=iter)
	}

	pred = array(0,dim=c(N,K))
	tr = array(dim=c(K,iter))

	for (i in 1:iter) {
		print(cat("Iterasjon ",i))
		train = rep(NaN,N)
		if (fraction > 0.0) {
			i_train = sample(which(is.finite(classes)), n)
		}
		else {
			for (j in 1:K) {
				i_train[(j-1)*n + 1:n] = sample(which(classes==j), n)
			}
		}


		train[i_train] = classes[i_train]
		ret_cross = kjorFullKlassifisering(pop, train, K, imgSize, beta=beta, jconv =
jconv, iter=iterRunde, adjustMuSigma=adjustMuSigma, adjustBeta=adjustBeta)
		if (adjustBeta==TRUE) {
			betaArr[i] = ret_cross$beta
		}
		for (j in 1:K) {
			pred[,j] = pred[,j] + (ret_cross$prediction==j)
		}
		for (j in 1:K) {
			ret_tr = countTreff(classes, ret_cross$prediction, class=j)
			tr[j,i] = (ret_tr$sensitivity + ret_tr$precition) / 2.0
		}

	}

	list(prediction=pred, treff=tr, beta=betaArr)
}



simulateImage <- function(fasit, K, mu_k, sigma_k) {
	library(mvtnorm)
```

```
      N = length(fasit)
      P = length(mu_k[1,])

      pop = array(dim=c(N,P))
      for (j in 1:K) {
            inx = which(fasit==j)
            pop[inx,] = rmvnorm(length(inx), mean = mu_k[j,], sigma=sigma_k[j,,])
      }

      for (i in 1:P) {
            pop[which(pop[,i]<0),i]=0
            pop[which(pop[,i]>256),i]=255
            pop[,i]=floor(pop[,i])
      }

      pop
}


computeGeneralizedMahalanobis <- function(mu_k, sigma_k) {
      K = length(mu_k[,1])
      omega = array(0,dim=c(K,K))
      for (i in 1:(K-1)) {
            for (j in (i+1):K) {
                  s = (sigma_k[i,,] + sigma_k[j,,])/2
                  d2 = mahalanobis(mu_k[j,], mu_k[i,], s)
                  g2 = 4*log(norm(s)/(sqrt(norm(sigma_k[i,,]))*sqrt(norm(sigma_k[j,,]))))
                  omega[i,j] = sqrt(d2+g2)
                  omega[j,i] = omega[i,j]
            }
      }
      omega
}
```